

document title/ titre du document

ESA / ESTEC  
INTERCHANGEABLE PEP  
(I-PEP)

TRANSPORT EXTENSIONS  
AND SESSION FRAMEWORK  
FOR SATELLITE  
COMMUNICATIONS:  
AIR INTERFACE SPECIFICATION

---

prepared by/ <i>préparé par</i>	SatLabs
Reference/ <i>référence</i>	
issue/ <i>édition</i>	1
revision/ <i>révision</i>	DRAFT
date of issue/ <i>date d'édition</i>	20 April 2005
status/ <i>état</i>	Draft
Document type/ <i>type de document</i>	Technical Note
Distribution/ <i>distribution</i>	

## A P P R O V A L

Title <i>titre</i>	issue 1 <i>issue</i>	revision <i>revision</i>	D R A F T
-----------------------	-------------------------	-----------------------------	-----------------------

author <i>auteur</i>	SatLabs PEP Working Group	date <i>date</i>	20 April 2005
-------------------------	---------------------------	---------------------	---------------

approved by <i>approuvé by</i>		date <i>date</i>	
-----------------------------------	--	---------------------	--

## C H A N G E L O G

reason for change / <i>raison du changement</i>	issue / <i>issue</i>	revision / <i>revision</i>	date / <i>date</i>
First DRAFT issue for circulation among SatLabs members	1	DRAFT	12 April 2005

## C H A N G E R E C O R D

Issue: 1 Revision: 1

reason for change / <i>raison du changement</i>	page(s) / <i>page(s)</i>	paragraph(s) / <i>paragraph(s)</i>

## T A B L E O F C O N T E N T S

<b>PREFACE .....</b>	<b>7</b>
<b>1 INTRODUCTION .....</b>	<b>8</b>
<b>2 SYSTEM OVERVIEW .....</b>	<b>10</b>
<b>3 PROTOCOL ARCHITECTURE.....</b>	<b>12</b>
3.1 CONSTITUENTS OF THE I-PEP SPECIFICATION .....	12
3.1.1 Terminology .....	14
3.1.2 Use of TCP and SCPS-TP .....	17
3.2 OVERVIEW OF PROTOCOL OPERATION.....	19
3.2.1 Transport Protocol .....	19
3.2.2 Session Protocol .....	21
3.3 I-PEP SERVICE "INTERFACES" .....	24
3.3.1 I-PEP Services Provided to Internet Applications.....	24
3.3.2 Communication Services required from the Network .....	25
<b>4 SCENARIO OVERVIEW.....</b>	<b>26</b>
4.1 SINGLE / MULTI-USER SCENARIO WITH I-PEP SERVER AT HUB .....	27
4.2 I-PEP SERVER AT ISP.....	29
4.3 MULTIPLE I-PEP SERVERS.....	30
<b>5 PROTOCOL SPECIFICATION: SCPS-TP.....</b>	<b>31</b>
5.1 TRANSPORT PROTOCOL SPEC: AN SCPS-TP "PROFILE" .....	31
5.1.1 General Updates to the SCPS-TP Secification .....	32
5.1.2 Specific IETF RFCs referenced by SCPS-TP .....	36
5.1.3 SCPS-TP Header Compression and Clarifications .....	38
5.1.4 I-PEP Congestion Control .....	41
5.1.5 Explicit Congestion Notification .....	42
5.2 TRANSPORT PROTOCOL EXTENSIONS.....	43
5.2.1 Introduction To Extensibility Mechanisms .....	43
5.2.2 SCPS-TP Capabilities Option Extension .....	43
5.2.3 The I-PEP Extended Option .....	50
5.2.4 TCP Option Data Notification .....	51
5.2.5 Session ID and Service Tag .....	53
5.2.6 Target Address.....	55
5.2.7 Vendor ID .....	56
5.2.8 I-PEP Option Space Extension .....	57

---

<b>6</b>	<b>SESSION LAYER FRAMEWORK .....</b>	<b>58</b>
6.1	ASSOCIATING I-PEP SESSIONS AND TRANSPORT CONNECTIONS.....	59
6.2	PROTOCOL OUTLINE .....	60
<b>7</b>	<b>CONFORMANCE SPECIFICATION .....</b>	<b>62</b>
7.1	MINIMAL FUNCTIONAL REQUIREMENTS AND TESTS .....	62
7.2	TEST TOOLS.....	63
7.3	INTEROPERABILITY SCENARIOS.....	64
7.3.1	Scenario 1: I-PEP client A and I-PEP server A .....	64
7.3.2	Scenario 2: I-PEP client A1 and I-PEP server A2 .....	64
7.3.3	Scenario 3: I-PEP client A and I-PEP server B .....	64
7.3.4	Scenario 4: I-PEP client B and I-PEP server A .....	65
7.3.5	Scenario 5: I-PEP clients A, B, C1, and C2 and I-PEP server C2 .....	65
7.3.6	Scenario 6: I-PEP clients A, B, C, and D and I-PEP servers C and D .....	65
7.4	I-PEP PICS USAGE .....	67
7.4.1	Introduction .....	67
7.4.2	NOTATION .....	67
7.4.3	I-PEP PICS STATEMENT .....	68
<b>8</b>	<b>SYSTEM ASPECTS .....</b>	<b>71</b>
8.1	GATEWAYS .....	71
8.1.1	Types of Gateways .....	71
8.1.2	Gateway Functionality.....	72
8.2	MANAGEMENT.....	78
<b>9</b>	<b>ADAPTABILITY AND EXTENSIBILITY .....</b>	<b>79</b>
9.1	IMPLICATIONS OF THE EXTENSIBILITY CONCEPT .....	81
9.2	EXAMPLES FOR VENDOR-SPECIFIC EXTENSIONS .....	83
9.2.1	Example 1: Application-Layer-Specific Extension.....	83
9.2.2	Example 2: Session Protocol Extension .....	83
9.2.3	Example 3: Parameter Extension .....	84
9.2.4	Example 4: Algorithm Extensions.....	84
9.2.5	Summary: Vendor Extension Signaling .....	85
<b>10</b>	<b>REFERENCES .....</b>	<b>88</b>

S

## PREFACE

The SatLabs Group [**SATLABS**] is an international, not-for-profit association whose members are committed to bringing the DVB-RCS standard to large-scale deployment by ensuring interoperability between DVB-RCS products.

The Group is working in the definition of a qualification and certification programme aimed at verifying compliance and interoperability of DVB-RCS terminals up to a certain level, defined as basic IP services. Basic IP services are defined as the ability to support forwarding of IP packets using protocols such as HTTP, Telnet, FTP and SNMP. Performance is, thus, not included in the definition of basic IP services.

Recognising that the market would like to see full operational interoperability, thus including, among other aspects, performance, and recognising that the performance of HTTP and FTP over satellite links is seriously compromised without the use of some kind of protocol enhancement, the Group agreed to select a common performance enhancement proxy (PEP) approach for DVB-RCS systems to ensure a higher level of interoperability.

The SatLabs Group has asked ESA to initiate a process of definition of a common PEP solution for the longer-term. This will consist of a future-proof standardised PEP air interface, including data formats and basic protocols. The objective is to allow the use of different interoperable PEP products in the client side (user terminal) and server side (hub station) that guarantee a minimum performance that is, however, satisfactory, for the end user from an application viewpoint.

This document contains a definition of an Interchangeable PEP definition [**I-PEP**]: it defines the basic procedures applied to and protocol messages exchanged over the air interface between satellite terminal and hub station.

This document makes extensive reference to the Transmission Control Protocol (TCP) and the SCPS-TP reliable transport protocol based upon TCP. The reader is assumed to be familiar with these specifications.

### Note on terminology

Performance Enhancing Proxy is used as defined in [**RFC 3135**] and currently considered as a “Transport Layer PEP” and “Split connections”.

The terms “Common PEP” and “I-PEP” are currently used interchangeably.

## 1 INTRODUCTION

Performance Enhancing Proxies (PEPs) became an integral part of today's "Fast Internet via Satellite" systems. This applies to unidirectional satellite systems, using some terrestrial return channel, as well as to bi-directional satellite communication systems.

Especially for bi-directional satellite communication systems—like DVB/RCS—the usage of PEP implementations became mandatory in order to meet even the basic user requirements for well performing and interactive Internet via satellite usage.

Over the past years, the functionality provided by PEPs has increased significantly: It started with basic TCP enhancement (mainly needed for fast bulk downloads only) and grew towards enhancing interactive application protocols like HTTP for typical "web surfing"—thereby confirming the status of HTTP as the by far most important application protocol used in today's Internet as well as used by the targeted DVB/RCS user groups. Typical additional functions of PEP implementations include e.g. compression, encryption, as well as access control.

DVB/RCS is today an emerging standard for bi-directional satellite communication systems. It is successfully implemented by many of the main vendors and has proven as a suitable and scalable system in large European installations.

After the successful evolution of the DVB/RCS standard, the focus of development and testing for DVB/RCS projects and DVB/RCS research moved towards guaranteeing the interoperability of different DVB/RCS systems from multiple vendors. Providing interoperable/interchangeable DVB/RCS systems is expected to boost the deployment of DVB/RCS even further—as this will allow the service providers and users to select exactly those components best fitting to their respective needs.

Currently, the option for the end users to freely select a mixture of DVB/RCS components from multiple vendors is complicated due to the fact that additional services and functions (beyond the scope of DVB/RCS standardisation) are to be implemented that span multiple DVB/RCS components. These additional functions include PEP implementations, that are typically a distributed implementation spanning the hub (server) as well as the client systems.

Therefore, the aim is to define a suitable Interchangeable PEP—"I-PEP"—protocol which, very similar to the scope of the DVB/RCS standard itself, should define the "air-interface" and thereby the communication between hub-side and client-side PEP instances. Implementations conforming to the "I-PEP" definition will allow using different PEP implementations on the client-side as well as on the hub-side and even allow the use of a complete mixture of different clients with different PEP implementations. This will provide users with the freedom to the end users to select DVB/RCS equipment fitting to their project needs.

By limiting the “I-PEP” definition to the air-interface “only” it shall be ensured that

- a) PEP implementations are compatible and interchangeable, while:
- b) the functions and implementation of the PEP implementations can easily be adapted to fit the rapidly changing technology of the Internet without requiring to update the “I-PEP” standard e.g. every some months;
- c) the implementations can easily be adapted to fit to e.g. hardware limits and operation system requirements for the corresponding DVB/RCS clients, thereby allowing “I-PEP” conforming implementations to be used on low cost DVB/RCS systems with e.g. very limited CPU power and memory as well as on high performance DVB/RCS client installations having lots of e.g. CPU and memory resources;
- d) still allowing to tightly couple certain local function (e.g. sending rate limitation in either direction, memory usage, buffer management) directly with the local hub or local DVB/RCS client systems irrespective of the client systems;
- e) not restricting a vendor’s choice on their products interact with Internet client and server components takes place (e.g. allowing for proxy-based enhancement solutions as well as for transparent ones); and
- f) not interfering at all with any hub- or client-local interfaces to certain vendor-specific or third party components (e.g. billing and accounting systems on the hub side or user interfaces on the client side).

The present revision of the I-PEP specification furthermore enables vendors to provide proprietary extensions so that they can benefit from broad interoperability (and approach new environments) and at the same time maintain their competitive strengths in homogenous deployments.

The focus on the air interface is highlighted in figure 1 below:

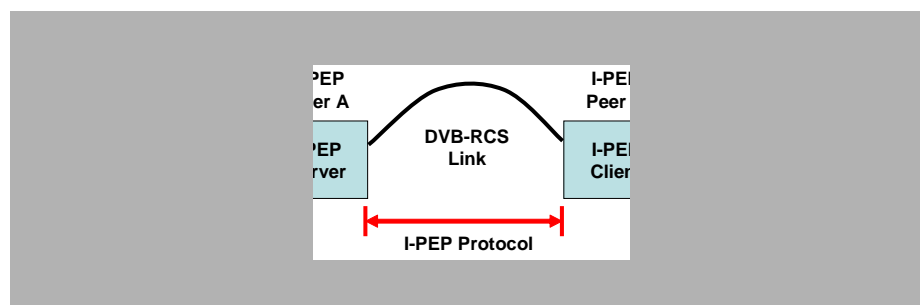


Figure 1: Focus of the I-PEP Protocol on the Air Interface

## 2 SYSTEM OVERVIEW

Figure 2 depicts the overall system scenario to outline the purpose of this specification: Two satellite service providers using equipment and software from different manufacturers offer DVB-RCS services for numerous customers—who in turn use a variety of receiver equipment and software. To enable such a heterogeneous scenario, the data formats and basic rules for information exchange across the satellite communication link need to be standardized.

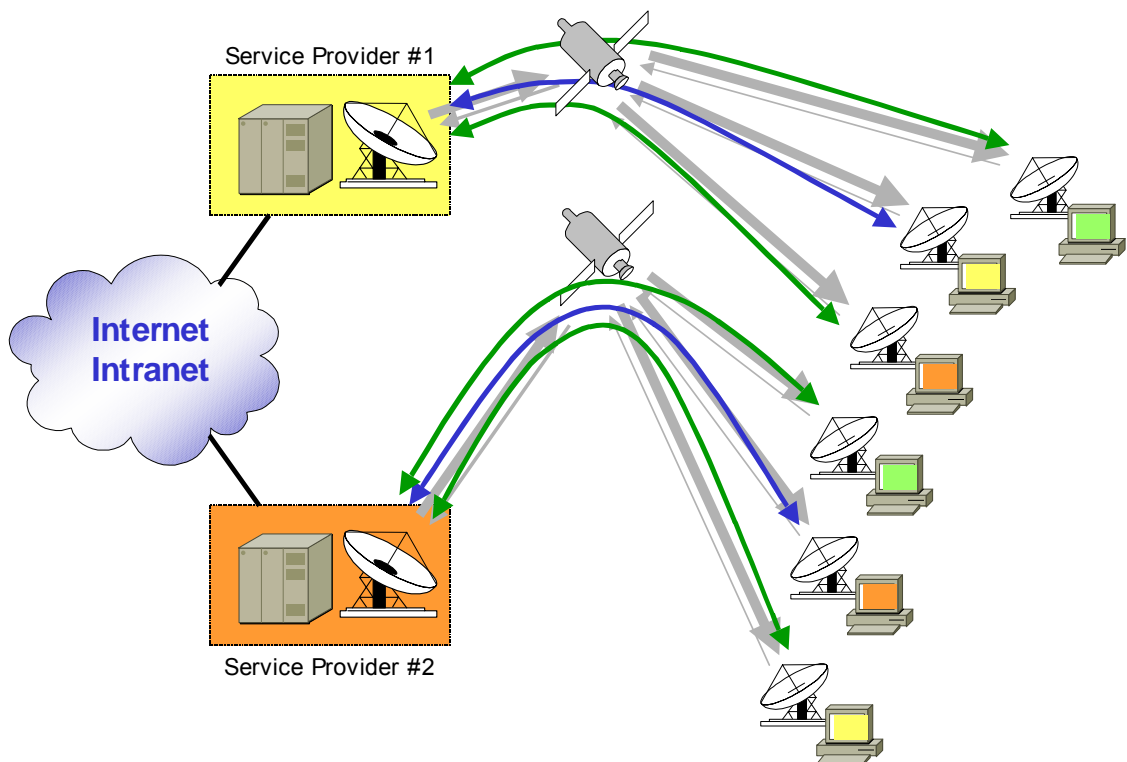


Figure 2: General overview

As shown in figure 2, two service providers are running servers from two different vendors, color-coded in yellow and orange. The customers utilize equipment from three different manufacturers, color-coded in yellow, orange, and green). All servers and clients can communicate at the level defined in this protocol specification (green arrows). Beyond this basic interoperability for efficient satellite communication, solutions from a single vendor on the client and server side can provide additional and/or further optimized services to their customers, benefiting from the vendor-specific extensibility of the present protocol specification (blue arrows). Finally, with this protocol specification strictly confined to the air interface, vendors may provide even more product

differentiation by intelligent proprietary algorithms on both the client and the server platforms: as local configuration, system integration, operation, and algorithms are left open.

The remainder of this specification defines the *Interchangeable PEP (I-PEP)* protocol and is structured as follows:

- Section 3: Architecture and Terminology
- Section 4: Scenario Overview
  - 1) Single user scenario: client PEP integrated with user PC
  - 2) Multi-user scenario: client PEP in external box shared by many entities
  - 3) PEP server at ISP (rather than satellite service provider)
- Section 5: Protocol Specification – a Profile of SCPS-TP
  - 1) Transport Protocol Spec: an SCPS-TP “profile”
  - 2) Transport Protocol Extensions (if any)
- Section 6: Session Protocol Framework
- Section 7: Conformance Specification
  - 1) Minimal functional requirements
  - 2) Torture tests
  - 3) Interoperability scenarios
  - 4) [I-PEP PICS]
- Section 8: System Aspects
  - 1) Gateway: Informal *functional* Definition
  - 2) Management
- Section 9: Adaptability and Extensibility
- Section 10: References

## 3 PROTOCOL ARCHITECTURE

### 3.1 Constituents of the I-PEP Specification

The I-PEP functional architecture assumes a split-connection approach with essentially two different roles of interest as depicted in Figure 3 below. The logical subdivision assumes an identifiable I-PEP server and a client capable of supporting the I-PEP protocol defined in this specification. Associated with the I-PEP client and server are an application client and server, respectively. The two I-PEP entities communicate via a DVB-RCS link, communication between application and I-PEP client on one side and application and I-PEP server on the other may be carried out via arbitrary local or wide-area networks using standard IP-based communication protocols such as TCP. In practice, application and I-PEP are likely to be co-located in the same LAN if not on the same physical device while application and I-PEP server are often likely to communicate across the wide-area Internet.

Note that the above distinction between client and server is artificially introduced to simplify distinguishing these entities in the description. The server parts are always considered to be located on the hub side of the DVB-RCS link, the respective clients on the DVB-RCS terminal side. The I-PEP protocol specification itself is symmetric in that both peers may have equal capabilities, may both invoke the same actions, etc. Similarly, the two peer applications (peers A and B) may take arbitrary roles: they can act both as servers or clients or as equal peers.

This document follows the above notion of client and server side for two reasons: 1) to follow the (most) common deployment of DVB-RCS and 2) to reflect the asymmetric nature that may be built into add-on protocols (such as service announcements) as defined in the context of the session (and management) layer in support of service location and load balancing.

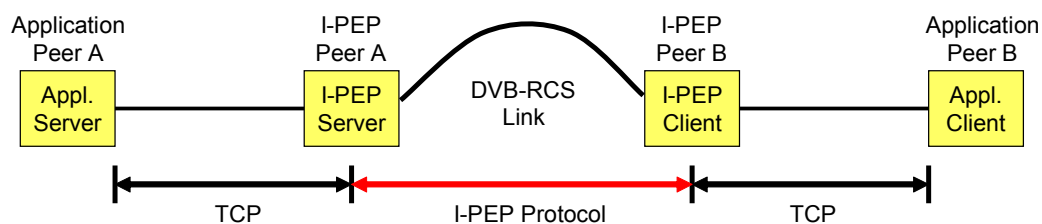


Figure 3: Basic I-PEP Components

As indicated above, the specification neither makes assumptions about the physical composition of the individual components nor about their communication relationships. Figure 4, shows possible integration structures and relationships between I-PEP and application entities. As depicted in

Figure 4a), a single peer of I-PEP entities may serve communications between an arbitrary number of application peers. Figure 4b) shows a setting in which each user (or application) device is served by exactly one I-PEP client; again, an arbitrary number of application servers is supported by a single server entity. However, the I-PEP server is replicated to provide for robustness and enable load balancing. While in these two cases, the I-PEP server is co-located with the uplink station, figure 4c) shows a setup in which this is not the case: the I-PEP server is placed in an external location (e.g., at an Internet service provider—as opposed to a satellite service provider). Figure 4c) also shows a complete integration of DVB terminal adapter, I-PEP client, and application client as may be found in fully embedded systems. All the above scenarios may be combined in an arbitrary fashion and, while not depicted in figure 4, the symmetry properties discussed above ensure that arbitrary applications may be run on the terminal side.

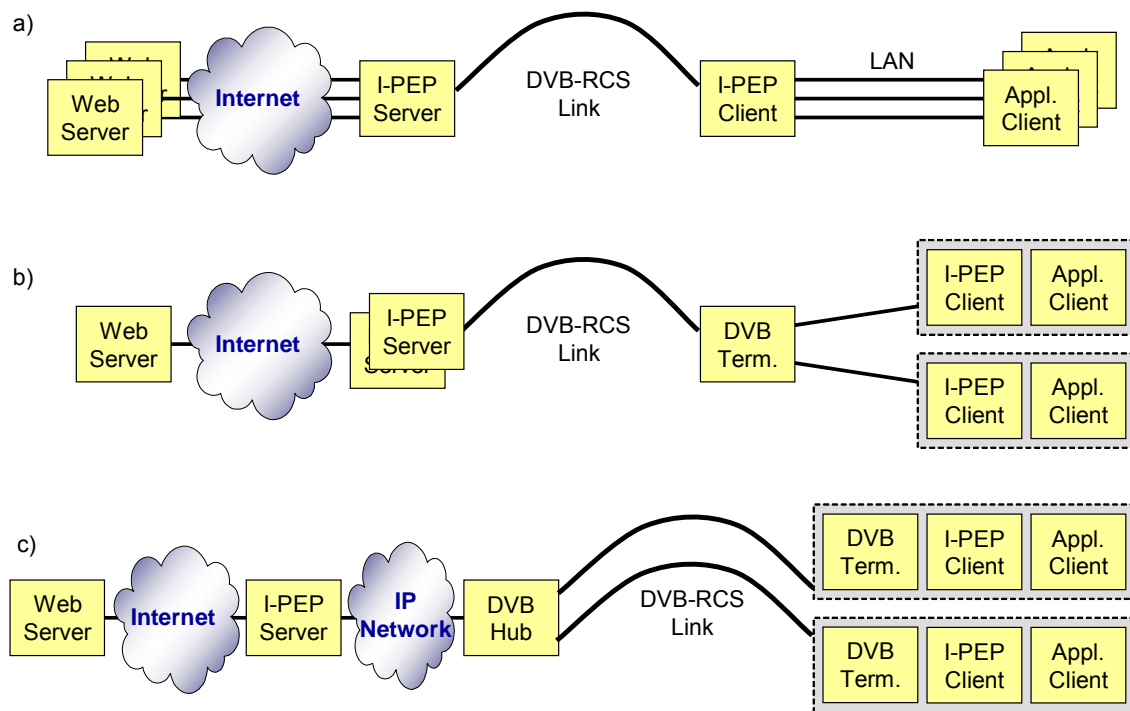


Figure 4: I-PEP Integration Examples

Finally, the I-PEP specification does not prescribe any particulars of the operation of the respective I-PEP components: whether the I-PEP entities perform some form of transparent capturing or whether it operates as (application-specific or generic) proxy or both, is up to the respective implementation. The I-PEP specification only specifies the necessary air interface to ensure interoperability between different systems. Similarly, policies and performance are entirely up to the respective implementation.

The following subsection provides a systematic overview of the I-PEP terminology used in this specification as partly already introduced above.

### 3.1.1 TERMINOLOGY

#### **I-PEP Peer**

An implementation of the I-PEP protocol specification as defined in this specification.

#### **I-PEP Server**

A system e.g. installed at or connected to a satellite service provider's premises that provides enhancement for satellite-based communications following this specification. An I-PEP server is capable of accepting incoming and creating outgoing I-PEP Connections.

From an I-PEP perspective, I-PEP server and client are symmetric I-PEP peers.

#### **I-PEP Session Server**

An I-PEP server is capable of accepting incoming I-PEP Sessions and may provide further functions relevant to service providers such as client authentication and accounting. The I-PEP Session server may but need not be the I-PEP server that will accept all client connections, but in the case that it is a distinct node then it must work in coordination with the associated I-PEP Servers.

#### **I-PEP Client**

A system e.g. installed / operating at a customer's / user's premises that provides enhancement for satellite-based communications following this specification. The client device acts on behalf of the user. It is capable of actively establishing outgoing I-PEP Sessions and I-PEP connections, accepting incoming I-PEP Connections, and is likely to provide some proxy stub functionality to communicate with existing local application clients (e.g. web browsers) to enhance their Internet access.

From an I-PEP perspective, I-PEP server and client are symmetric I-PEP peers.

#### **I-PEP Protocol**

The satellite communication protocol providing efficient and reliable communication via satellite links between a client device and a server device. The I-PEP protocol consists of a transport protocol heavily based on TCP and modified/augmented by SCPS-TP as well as a session protocol comprising several optional additions to support service and session management. Future extensions may add application-specific functionality (such as HTTP Prefetching) to the I-PEP protocol.

**I-PEP session message**

This term refers to the atomic information units exchanged between I-PEP peers supporting the I-PEP Session protocol.

**HTTP Prefetching**

A set of extensions to the regular HTTP protocol to be used in conjunction with the I-PEP protocol in order to provide further performance enhancement.

With HTTP prefetching, the server and/or client device receives a request for a web page (i.e., a web resource) via HTTP and an I-PEP Connection from the web browser (via the client device) and act(s) upon it. The HTTP prefetching action incurs that the original request may be forwarded to the server, and requests for additional web resources may be generated by the client and/or the server device *before* a corresponding request is received from the web browser so that the client and/or the server device may obtain additional web resources before the web browser asks for these. This approach allows client and/or server device to be more responsive to (future) requests from the web browser and thus may reduce the overhead incurred by requests and responses repeatedly traversing the network end-to-end and thus may significantly increase interactivity and thus the perceived performance for users.

**I-PEP Transport Connection**

An I-PEP Connection is the logical representation of an Application TCP Connection from a client to a server between the I-PEP peers. That is, there is a one-to-one mapping between every Application TCP Connection from an application client via the I-PEP protocol to an application server and an I-PEP Connection.

An I-PEP Connection provides bi-directional data exchange in a reliable and flow-controlled manner. The basic service is unmodified data transmission but data may also be compressed. On top of one or more I-PEP Connections, additional services may be provided. This specification, however, provides only the framework for offering such services but does not specify any of these.

It should be noted that additional I-PEP Connections may be used between client and server device that are not related to Application TCP Connections. These may be established dynamically by either of the I-PEP peers whenever deemed necessary, e.g. for HTTP prefetching.

**I-PEP Session**

A “meta” communication relationship between two communicating peers (such as a user’s I-PEP client and satellite provider’s I-PEP server). This is an optional relationship that may—but need not—be established between the two entities using the session signaling. If used, there will often be exactly one I-PEP session between a client device and a server device.

An I-PEP session provides a shared context for an extended period of time during which any number of I-PEP transport connections are established, used, and torn down between the peers. The shared context may contain information for initial configuration of I-PEP transport connections, provide capability information about the peers, be used to initialize security functions (e.g., encryption), etc.

### **Application TCP Connection**

A TCP connection intended to connect an application client and its server (or two peers) end-to-end. Due to the connection splitting approach applied with the I-PEP, the connection is usually established between an application client (e.g. a web browser) and the I-PEP client and a corresponding one between the I-PEP server and the respective application server. In certain cases, e.g., if additional application layer services (such as HTTP prefetching) are in use, a (client-side) Application TCP Connection may already be terminated and responded to by the I-PEP client or server.

As can be seen from figure 1, the TCP connection that would normally run directly between the application client and the application server is split into three pieces, with TCP being replaced by the I-PEP protocol—and application-specific extensions such as HTTP prefetching (if applicable)—on the middle piece running via the satellite link.

For every Application TCP Connection coming in from an application client at either peer, a corresponding I-PEP Connection may be created (operating independently of the other I-PEP Connections) if necessary. If so, subsequently, a corresponding Application TCP Connection may be established at the remote side from the peer to the respective application server if needed. Then, data exchanged between application client and application server is forwarded end-to-end. Separate modules, such as HTTP prefetching, may provide additional services beyond plain data forwarding; in particular, they may terminate an Application TCP Connection at either the I-PEP client or server and respond to the application client locally.

### **(I-PEP) Transport Protocol**

The protocol at the transport layer used between I-PEP peers to provide reliable, flow-controlled, and rate- or congestion-controlled data transmission. The I-PEP transport protocol is derived from TCP, following selected parts of the SCPS-TP specification, and additions / modifications as defined in this specification.

### **(I-PEP) Session Protocol**

A minimal control protocol defined in a separate document. A framework for the I-PEP Session Protocol is outlined in section 6 of this specification. This protocol is optional. It may be used to inform I-PEP clients about available I-PEP servers and their capabilities and to negotiate parameters of an I-PEP session.

### **Transparent capturing**

Transparent capturing denotes one possible way of intercepting TCP connections for the purpose of introducing performance enhancement functions (another option is proxy-based

operation as defined below). With transparent capturing, the application initiating the TCP connection need not be modified or reconfigured as the interception takes place on the wire *transparent* to the application.

Transparent capturing allows for two different modes of operation regarding end-to-end semantics at each I-PEP peer (note that the same options are available for connection teardown via the FIN-ACK handshake): 1) Connection establishment may preserve **end-to-end** semantics in that the connection setup is not confirmed but by the ultimate destination so that an incoming SYN packet is not acted upon immediately but, instead, a corresponding SYN packet is generated towards the ultimate destination. 2) The SYN packet may already responded to by the I-PEP peer close to the initiator of the TCP connection (**local spoofing**), to the effect that the TCP connection setup completes immediately with the local I-PEP peer. The end-to-end establishment follows suit but may lead to an immediate disconnection if the ultimate destination cannot be reached for some reason. Note that this decision may be taken individually by each of the (two) involved I-PEP peers resulting in four combinations only one of which preserves the end-to-end semantics of the TCP connection establishment.

### Proxy-based operation

In contrast to transparent capturing, proxy-based operation refers to communication setups in which TCP connection from the application peer is explicitly targeted at its corresponding I-PEP peer (e.g. from an application client to the I-PEP client). This means that the capturing is not entirely transparent to the application and implies that the application TCP connection is terminated at the I-PEP peer and that connection setup does not preserve end-to-end semantics. Proxy-based operation may be used with application protocols that explicitly support proxies (such as HTTP) or that support SOCKS or other control protocols. Furthermore, arbitrary applications are supported that connect to a predefined target (such as a mail or file server).

### 3.1.2 USE OF TCP AND SCPS-TP

As discussed above, the I-PEP transport protocol is used to efficiently carry data between two I-PEP peers. To avoid defining a completely new protocol from scratch and thereby to allow for achieving interoperability even with non-PEP peers, *the* standard reliable transport protocol in the Internet is taken as a basis: TCP (RFC 793, RFC 1122, RFC 2581, among others).

As it has been repeatedly proven, TCP is not well suited for communication across network paths that a) exhibit a high bandwidth  $\times$  delay product or may b) show a significant fraction of non-congestion related packet losses. Satellite links by definition fall into category a) and, depending on setup and environmental conditions, may also show characteristics of b).

These—well known and documented—deficiencies of TCP have been addressed in the past in the IETF (in general) as well as by other (standardization) bodies. While the IETF has produced numerous general purpose TCP extensions—e.g. to support large windows and selective

acknowledgements, among many others—other groups have focused on dedicated environments, such as satellite communications investigated by the Consultative Committee for Space Data Systems (CCSDS) that has produced an extension set to and profile of TCP termed SCPS-TP in 1999. SCPS-TP addresses parts of the issues with TCP for satellites and maintains backward compatibility with vanilla TCP.

The I-PEP specification builds on top of TCP, SCPS-TP, and recent TCP extensions and creates a protocol suitable for satellite communication via DVB-RCS while maintaining backward compatibility with TCP and SCPS-TP.

Taking SCPS-TP as a starting point, the I-PEP eliminates all those options that are not meaningful for reliable communications via DVB-RCS (including those that can be easily accomplished independently, e.g., by just sending UDP datagrams), creating a specific I-PEP profile from the available SCPS-TP capabilities. For example, the best effort transport service (BETS) from SCPS-TP is not needed, congestion control rules need to be adapted, and several parameters and options deserve revisiting.

Functions that are deemed relevant for satellite communications but not yet included in SCPS-TP are added in this specification. The extensions to the transport protocol are kept to a minimum and other (e.g., session-related) functionality is pushed into a separate protocol orthogonal to the transport.

## 3.2 Overview of Protocol Operation

The I-PEP protocol consists of two parts: the transport and the session protocol. The transport protocol is mandatory and is therefore outlined first. The session protocol is optional and is addressed subsequently.

The overall operation consists of the following interactions (that need not necessarily be carried out in that order):

- 1) I-PEP Server Location
- 2) Session setup and minimal feature negotiation (optionally including authentication)
- 3) Establishment of I-PEP transport connections
- 4) Data transfer across I-PEP transport connections
- 5) Teardown of I-PEP transport connections
- 6) Dynamic session reconfiguration
- 7) Session teardown

Out of these “interactions”, only 3) – 5) are mandatory and covered by the transport protocol. All others are optional.

### 3.2.1 TRANSPORT PROTOCOL

The transport protocol operation resembles that of TCP.

Either I-PEP peer may initiate an I-PEP transport connection—e.g. when an incoming application TCP connection is noticed and shall be forwarded to a remote peer. The interaction between the client or server applications and the I-PEP device is not related to the air interface, and therefore not specified by the I-PEP standard. How the ultimate target address is determined is implementation dependent: transparent capturing, interpretation of application protocols, explicit signaling, and local configuration are possible alternatives, among others. The end destination IP address of the outgoing SYN packet MAY be the ultimate destination address of the TCP connection or it MAY be the address of the I-PEP peer. To exercise the latter case, the initiating I-PEP SHOULD have prior positive knowledge (e.g., by session layer negotiation or configuration) that its I-PEP peer has alternative means available to determine the ultimate target address (e.g., application protocol interpretation, explicit signaling, or static configuration as described above).

In either case, the I-PEP server MUST intercept the I-PEP transport connection (i.e., terminate the I-PEP transport segment across the DVB-RCS link).

Editor’s note: An I-PEP peer will notice an incoming I-PEP transport connection by means of the I-PEP-specific TCP options in the SYN packet, regardless of the destination address carried in the packet. If the destination IP address is identical to the intercepting I-PEP peer, it is its responsibility to terminate the I-PEP transport connection and perform further actions, e.g., based upon the application-layer information carried in I-PEP transport header options or in the connection (e.g.,

processing HTTP requests targeted at a proxy) and deduce the ultimate destination address from the application layer information. Otherwise, the IP address is considered to belong to the ultimate destination and, in this case, this IP address **SHOULD** be used for connection setup to the respective application peer (unless configured otherwise, e.g., to contact a web cache).

SYN packets are used to establish the I-PEP transport connection. TCP options are used to indicate this being an I-PEP connection. TCP options are also used for—naturally limited—negotiation of connection-specific parameters (if any) and to optionally bind a new I-PEP transport connection to an existing session context (see below).

Negotiation of the I-PEP transport options and connection establishment are completed with the receiving peer completing the SYN-ACK handshake. After this, the transport connection is available for use.

Data transmission is achieved by an I-PEP peer by forwarding data received from an application TCP connection to the corresponding I-PEP connection and vice versa. Data received from the application TCP connection is locally acknowledged by the I-PEP peer before being sent across the I-PEP connection. Data transmission across the I-PEP transport connection should be congestion-controlled or rate-controlled as suitable for the respective deployment environment. Implementation of flow control across this sequence of connections is implementation-dependent—but an implementation of an I-PEP peer has to ensure that no data is lost between the establishment and teardown of the application TCP connection.

Teardown of an I-PEP connection is initiated by either of the application peers: if a FIN (or RST) packet is received for an application TCP connection the corresponding I-PEP connection is torn down as well, using a FIN or RST.

For I-PEP connection setup, the I-PEP connection may operate in two modes: hop-by-hop and end-to-end. In the end-to-end case, an I-PEP peer may decide not to act upon an incoming SYN packet, send on a corresponding one, and wait for the remote side—and thus the other I-PEP peer or the final target of the connection—to acknowledge the SYN packet before sending an ACK to the initiator of the application TCP connection. Alternatively, in the hop-by-hop case, the I-PEP peer may complete the SYN-ACK handshake for the incoming application TCP connection locally in parallel to initiating the I-PEP connection to the remote peer. The same considerations apply to the I-PEP receiving an incoming I-PEP connection and opening an outgoing application TCP connection to the final target. This yields four different combinations, and true end-to-end semantics for the establishment of an Application TCP Connection is only achieved if both I-PEP peers operate in end-to-end mode.

TCP options are *not* carried end-to-end as it is the intention of the I-PEP transport protocol to perform connection splitting. The I-PEP transport requires considerable option space for negotiation between the two I-PEP peers, leaving little room for end-to-end negotiation. In fact, most TCP options are relevant for hop-by-hop operation in the split connection scenario anyway. The I-PEP peers are expected to be conforming to standard TCP when communicating with their respective application peers and perform proper option negotiation. An I-PEP implementation **SHOULD** provide for TCP timestamps, large windows, and selective acknowledgements (SACK), and **MAY** implement further TCP options.

Data markings by the TCP URGENT pointer MAY but need not be preserved end-to-end. Note that, in practice, virtually none of today's widespread applications make use of this feature (and they are expected to work without the urgent pointer) so that this capability is likely to provide little benefit while adding implementation complexity.

It is local matter of an I-PEP implementation how to handle disparate TCP windows, how to deal with ECN and congestion, how to perform buffering, etc. It is also a local matter of an I-PEP implementation how to map incoming TCP segments to outgoing ones: in particular, it is not possible to guarantee that (the content of) each incoming TCP packet will be mapped to exactly one outgoing one. Different MTU sizes and different options in the TCP header may cause data to be split or allow for aggregation. Note that this is in line with the TCP service of reliable data stream delivery that makes no assumptions of how TCP user data is distributed across individual IP packets.

The end-to-end vs. hop-by-hop considerations also apply to TCP connection teardown (FIN – ACK) and reset (RST). For a split end-to-end Application TCP Connection, an I-PEP implementation SHOULD NOT send FIN packets by itself, unless an error condition is observed. Instead, it SHOULD only send FIN packets upon receipt of a FIN packet from an application peer. It is up to the I-PEP implementation to decide whether or not to ACK the FIN packet immediately or whether to forward a corresponding FIN packet to and wait for an acknowledgement from the remote I-PEP or application peer. If an I-PEP peer decides to wait for a remote acknowledgement, it SHOULD attempt to forward all outstanding data prior to forwarding the FIN packet and it SHOULD NOT send a FIN packet in the opposite direction on its own before having received one from the remote peer. Otherwise, it is up to the I-PEP peer to decide whether or not to discard locally buffered data before forwarding the FIN packet and whether or not to locally issue a FIN packet in the opposite direction.

Processing and/or forwarding of RST packets is entirely up to the I-PEP implementation. Since RST is usually used to indicate a protocol error, delivery of buffered data is NOT RECOMMENDED.

Neither of these modes of operations is preferred by this specification, and each I-PEP peer MAY take the decision independently (leading to four different combinations). End-to-end semantics for the setup is only achieved, however, if both I-PEP peers decide to operate in end-to-end mode.

### 3.2.2 SESSION PROTOCOL

The I-PEP session protocol offers several functions to organize the interaction between I-PEP servers and their I-PEP clients. All of these functions are optional.

- I-PEP Session Server Location

Initially, the I-PEP client needs to determine a (suitable) server. Information about available services may be made available via appropriate service announcements via the satellite. Alternatively, a DNS name defined by the service provider (e.g.,

ipep.dvbrcs.service-provider.net) may be used for dynamic server lookup: this DNS name MAY be registered with a DNS server as A/AAAA record; the DNS address(es) MAY point to the I-PEP server or the I-PEP server MAY intercept the TCP connections targeted at one of these addresses. The I-PEP server MAY also intercept DNS queries for this name and return an appropriate (set of) IP address(es). Finally, an I-PEP server may be statically configured or otherwise provided by external means.

On I-PEP client side processing, a service announcement (if available) SHOULD take precedence over a DNS lookup (of a configured name). Both SHOULD take precedence over a statically configured IP address.

Also, if an I-PEP client communicates with an I-PEP server, the I-PEP server may provide information about additional server devices that may be contacted (e.g. for load balancing or failure recovery).

The I-PEP client selects one of the proposed I-PEP servers. If later session or connection setup fails, the I-PEP client chooses another one.

- I-PEP Session Establishment

After an I-PEP client has chosen an I-PEP server, the client MAY contact the server to set up an I-PEP Session. An I-PEP Session is essentially a shared context for future communications between the two devices. At a minimum, the I-PEP Session is used to perform basic capability negotiation between the two peers and to establish a few operational parameters that can be shared across transport connections. I-PEP client and server device identify themselves in terms of their respective manufacturer and the I-PEP protocol versions they are supporting so that the devices know which features they may rely upon during later communications.

The I-PEP session MAY also be used to authenticate a client and to negotiate a shared secret for future confidential communications and other parameters. The I-PEP session can be used to negotiate other services, such as billing.

An I-PEP Session is identified by a Session ID.

An I-PEP Session shall not be used to carry user's payload. This is to avoid breaking the relationship that for each application TCP connection exactly one I-PEP connection exists.

- I-PEP Session Termination

The I-PEP Session MAY be terminated by the client or the server device through an explicit teardown handshake or implicitly when no longer used. Session termination MAY also stop accounting for the respective client device on the server side.

- I-PEP Session Maintenance

While an I-PEP Session is established, occasional control messages are exchanged between client and server to supervise the overall system operation and possibly update session context parameters. The overhead of these control messages is kept to a minimum in order to not waste precious satellite link capacity.

### 3.3 I-PEP Service “Interfaces”

#### 3.3.1 I-PEP SERVICES PROVIDED TO INTERNET APPLICATIONS

The I-PEP Protocol offers a communication channel between a client device and a server device, intended to efficiently support information exchange between a client application and a server application, e.g. when accessing the Internet.

The I-PEP transport is based upon TCP and offers the following services:

- reliable communication (ordered, error-protected, flow-controlled delivery of information);
- bandwidth efficient communication by minimizing protocol overhead; and
- high performance communication through reduced number of round-trips and specifically adapted congestion / rate control for high utilization of the satellite capacity—which may be adapted per implementation.

In addition, the session protocol provides minimal support for:

- server location and minimal feature negotiation between peers;
- network properties which should be considered by the I-PEP (for example, maximum uplink capability, minimal bandwidth allocation, permitted HTTP Acceleration methods).

Optional support may be included within the session protocol:

- authentication of the client device (and thus the user);
- enabling secure communication (setting up an encryption context for the contents exchanged via the satellite)

Finally, the specification provides hooks to later add extensions for application-specific support such as

- HTTP prefetching; and
- enhancements for further application protocols.

Proprietary extensions to the I-PEP protocol may provide further services or differentiate in the way and performance they offer the above services.

The service interface is accessed via standard application TCP connections. An API is not defined.

### 3.3.2 COMMUNICATION SERVICES REQUIRED FROM THE NETWORK

To provide these services, the I-PEP protocol relies upon a few lower layer communication functions to be provided by some IP-based network and the local protocol stack:

- bidirectional IPv4 unicast communications;
- appropriate address resolution and IP routing to the clients;
- optionally, the User Datagram Protocol (UDP) as the basis for session layer protocol announcements to provide addressing (port numbers) and a checksum and the optional use of IPv4 multicasting.

Proprietary extensions to the I-PEP protocol may utilize further IP and transport layer functionality for further vendor differentiation.

## 4 SCENARIO OVERVIEW

This section outlines three application scenarios for the I-PEP protocol that were identified from presentation by and discussions with SatLabs members as well as with ESA. The differentiation can be made on the client (DCB-RCS terminal) side as well as on the server (DVB-RCS hub) side.

On the client (terminal) side, we focus on a scenario in which the I-PEP client is running in a stand-alone box (with or without the DVB-RCS modem in the same box) while the end user device (PC, laptop, etc.) is a physically separate entity (section 4.1): In this case, a single user or a group of users may be connected to the I-PEP client box, e.g. via some local area network.

Note that an I-PEP client may also be integrated with the end user device; however, this is not the focus of this specification.

On the server side, two settings are conceivable:

- Integrated I-PEP server (I-PEP server co-located to with the DVB-RCS hub (which is assumed in the above scenarios) or
- Remote I-PEP server (I-PEP server topologically distant to DVB-RCS hub, section 4.2).

Finally, there may be more than a single I-PEP server and IP layer routing and/or I-PEP client configuration may determine which I-PEP server is used for a particular transport connection (section 4.3).

The following three subsections describe these scenarios.

## 4.1 Single / Multi-user scenario with I-PEP Server at Hub

In the simplest case, i.e. the single user scenario, there is a clear one-to-one mapping between users (not necessarily applications or application instances though) and I-PEP clients. This reflects the typical home user or home office scenario.

This specification is targeted at I-PEP clients implemented in a piece of equipment separate from the user's end device. The I-PEP client may be integrated with the DVB terminal as shown in the topmost part of figure 5 on the right, or it may be a stand-alone entity separate from both end user's device and DVB terminal (as shown in the middle part of figure 5 on the right). Due to the physical separation of the user's end system and the I-PEP client, the interaction between the two is restricted to (standardized) protocol exchanges. The I-PEP specification should also be applicable in cases where the I-PEP client and the end user device are integrated.

The multi-user scenario expands beyond the single user variant in that several application clients are served by the same I-PEP client (which implies some implicit trust relationship between the user and the service provider). This is shown in the lower part of figure 5 on the right. As in the above two cases, the I-PEP implementation is completely decoupled from the user device and the interaction between the two is restricted to (standardized IP-based) protocol exchanges.

The scenario reflects the typical small office, branch office, or hot-spot scenario. Of course this scenario and the ones described above may be mixed.

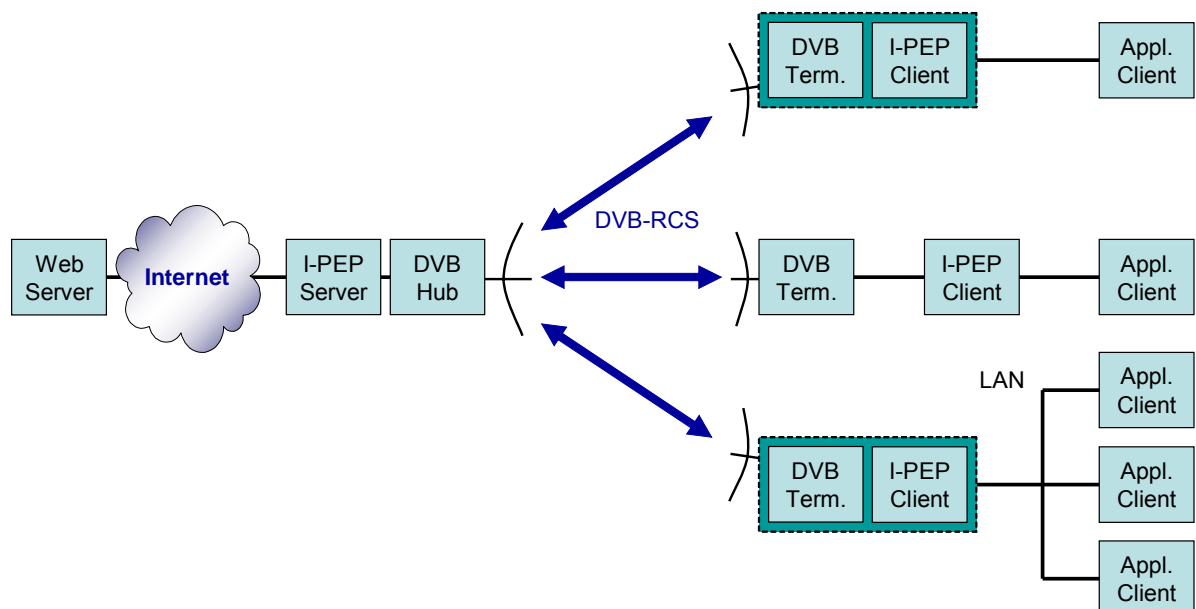


Figure 5: Single and multi-user scenario with I-PEP server co-located with the hub

In both scenarios, the I-PEP server is co-located with the DVB-RCS hub and we assume that both operate in the same address range so that no address transformation is required and I-PEP server and DVB-RCS may be assumed to reside within the same trust domain. This also eliminates the need for managing the “contribution” link from the (Internet) service provider to the satellite service provider.

## 4.2 I-PEP server at ISP

The scenarios above have, for simplicity, assumed that the I-PEP server will be co-located with the DVB hub station (which implies that the satellite service provider either operates the I-PEP or provides at least hosting facilities for the respective system components). In a different setting, the I-PEP server is external to the hub station, motivating two different setups:

- the I-PEP server may be run by a separate Internet Service Provider (ISP) on behalf of many users or
- the I-PEP server may be operated by an enterprise on its own behalf.

In both cases, of course, multiple external I-PEP servers running independently in different locations are conceivable. This is shown in figure 6.

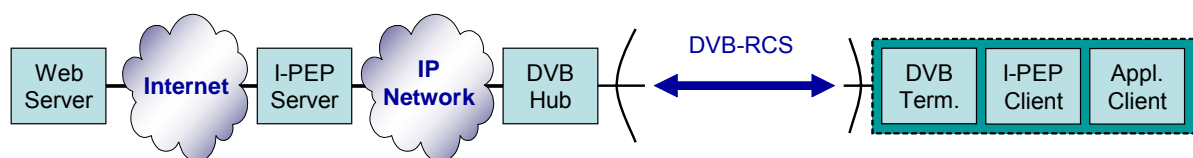


Figure 6: Server side with I-PEP server located at ISP (rather than satellite service provider)

The primary differences between this setting and the above two are as follows:

- The communication link between I-PEP server and DVB hub now extends through a wide area network that is not trusted and the satellite service provider may also not be trusted.
- Addressing schemes for I-PEP clients and DVB-RCS terminals may be controlled by two different institutions.
- A close clustering of multiple I-PEP servers is no longer possible since they may be located in different places.
- Different I-PEPs may be operated by different service providers.

QoS aspects on this feed link can be addressed by proper dimensioning and traffic/QoS engineering and do not require further consideration.

### 4.3 Multiple I-PEP Servers

In this scenario, there is no longer a single “centralized” I-PEP server. Instead, multiple I-PEP servers are used: either because multiple (enhancement) service providers are used or because performance enhancement is managed directly between user sites. Thereby, it becomes possible to accelerate traffic between directly communicating DVB-RCS terminals directly, using the flexibility offered by SCPS-TP.

While uncommon in Internet access via satellite installations for home users, SOHO environments, and the like, this scenario may be implemented in conjunction with (virtual) enterprise networks.

Note that, when I-PEP terminals or user sites run their own I-PEP peers and communicate directly, there is no strict separation into an I-PEP server and I-PEP client role. The figure below keeps this terminology just for consistency.

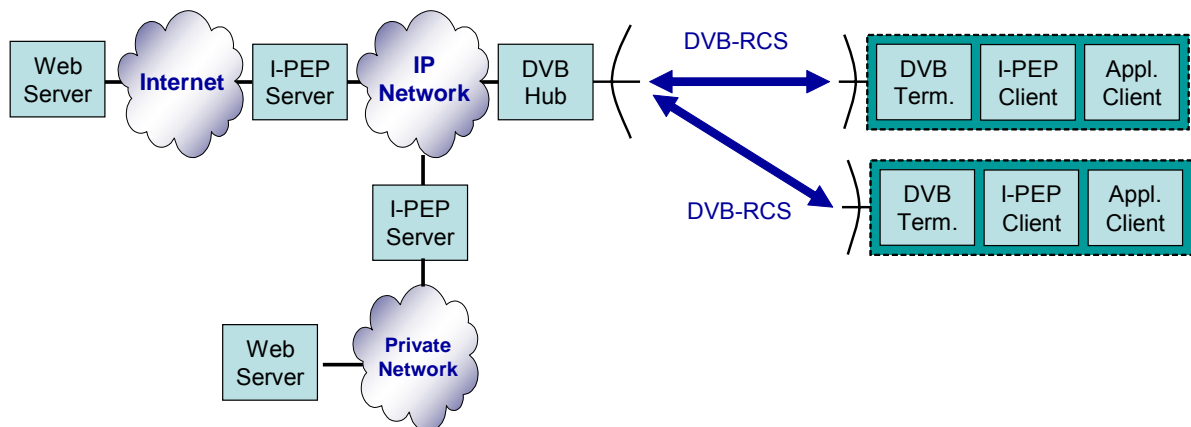


Figure 7: I-PEP clients accessing multiple I-PEP servers

In the scenario depicted in figure 7, there are multiple hub I-PEP servers, e.g., one located for general use Internet access and another at ISP premises. This scenario may be useful in cases where a remote terminal, through local implementation, has an encrypted tunnel for enterprise communications as well as a direct Internet connection for general communications, and I-PEP acceleration is required for both and can only be done independently.

## 5 PROTOCOL SPECIFICATION: SCPS-TP

This section defines which parts of SCPS-TP to use and which not in an I-PEP implementation and outlines implementation options for interpreting SCPS-TP. It also provides restrictions as well as extensions to the fields and parameter sets of SCPS-TP. Section 5.1 addresses how SCPS-TP is used and section 5.2 defines extensions beyond SCPS-TP.

### *5.1 Transport Protocol Spec: An SCPS-TP “Profile”*

This subsection specifies the details of the usage of SCPS-TP for the purpose of performance enhancement via bi-directional satellites. SCPS-TP as defined in CCSDS 714.0-B-1 (Blue Book, May 1999) will be used as a basis, particularly the detailed specifications in sections 3 through 6 of the SCPS-TP document

- the SCPS-TP features mandatory for I-PEP are stated,
- the SCPS-TP features optional for I-PEP are listed,
- modifications to SCPS-TP are fully defined,
- extensions to SCPS-TP are fully specified, and
- parts of SCPS-TP not applicable are identified.

Where extensions are needed, those are specified in section 5.1.5 below.

Updates to the SCPS-TP specification are ongoing. These updates are first captured as “CCSDS Concept Papers” before integration into an updated SCPS-TP document known as a “Pink Sheet”. I-PEP implementers are encouraged to review these concept papers, and consider implementation of mature updates if they may be of benefit.

<http://public.ccsds.org/publications/CCSDSConceptPapers.aspx>

### 5.1.1 GENERAL UPDATES TO THE SCPS-TP SPECIFICATION

SCPS-TP was designed as specification including use in spacecraft and tactical environments. These portions are not applicable or useful for the purposes of the I-PEP specification.

The SCPS-TP specification for the use of TCP (as defined in section 3 of the SCPS-TP specification) shall be applied as described in the following:

#### Section 3.1 [applicable]: Relationship between SCPS-TP and TCP

TCP shall be used as the basis.

#### Section 3.2.1 [modified]: Initial Sequence Number Selection

Initial sequence numbers SHOULD be random.

#### Section 3.2.2 [not applicable]: Precedence Handling

Precedence handling can be ignored as I-PEP is implemented on top of the standard Internet Protocol, not the SCPS Network Protocol (NP).

#### Section 3.2.3 [extended]: Negotiation of SCPS-TP Capabilities

Support for the SCPS Capabilities Option format is required.

Extensions to the SCPS-TP capability option are defined to meet I-PEP requirements. These extensions are defined in section 5.2.2 below.

#### Section 3.3 [not applicable]: Data Transfer

This part is not relevant and MUST be ignored in its entirety, as an I-PEP considers only reliable TCP transport.

#### Section 3.4.1 [applicable]: Transmission Timeout

Transmission timeout SHOULD be dynamically determined. A static value SHOULD NOT be used except for initial setting. The initial value chosen by an implementation SHOULD take into account the intended application area (i.e., geostationary satellite communications) and consequently SHOULD be in the order of the expected RTT.

#### Section 3.4.2 [modified/extended]: Congestion Control and Corruption

The congestion control requirements (3.4.2.1 through 3.4.2.3) are extended as described in section 5.1.4 below.

Use of 3.4.2.4 is optional.

DVB-RCS links do not pass corrupted data to IP endpoints. Therefore, the Corruption Experienced (“CE”) Option proposed by SCPS-TP in sections 3.4.2.5 and 3.4.2.7.3 is not applicable.

Use of 3.4.2.7.4 is optional.

### Section 3.4.3 [modified]: Link Outages

Link outage support of section 3.4.3 is an implementation specific decision and is not specifically addressed by the I-PEP specification.

### Section 3.5 [applicable]: Selective Negative Acknowledgements (SNACK)

Support for SNACK is mandatory as per SCPS-TP. Support for the long form SNACK (bit vector) is optional.

Note: SNACK is more bit efficient than SACK, both for each connection negotiation (it is done within the SCPS-TP Capabilities Option), as well as for transmission of SNACKs. It can also be quite responsive to burst loss compared to pure SACK. Therefore, SNACK is required for implementation.

The “Long” SNACK, as determined through simulation and analysis, is useful only within a relatively narrow bandwidth-delay product range and requires considerable buffering of data before sending an effective acknowledgement. Therefore, the 3.5.2.5 SNACK Bit-Vector option is not recommended for implementation but permitted.

Regular TCP SACK [RFC 2018] MAY be used but the use of SNACK SHOULD be given preference.

### Section 3.6 [modified]: Header Compression

Header compression MAY be supported; if supported, the modifications specified in section 5.1.3 apply. Header compression is useful for reducing return channel overhead.

### Section 3.7 [not applicable]: Multiple Transmissions for Forward Error Correction

Multiple Transmissions for forward error correction is a function primarily designed for links other than Satellite communications and is not recommended for implementation.

Other (more efficient) FEC schemes MAY be used as vendor-specific extensions provided that the I-PEP transport transmission rate is congestion controlled.

#### Section 4 [not applicable]: User Datagram Protocol Extension

I-PEP support is specified for TCP only.

#### Section 5 [optional]: Management Information Base

The management information base as specified by SCPS-TP is not required for implementation. There are no MIB requirements currently defined by the I-PEP standard.

#### Section 6 [modified]: Conformance Requirements

In general terms, the following considerations apply:

##### Section 6.2.1 [modified]: Major Capabilities

- Only TCP is required, UDP is not (nor are any functions pertaining to unreliable transport).
- The basic requirements for Internet hosts [RFC 1122] and TCP [RFC 793] apply unless explicitly specified otherwise in this document.

##### Section 6.2.2 [modified]: Mission-specific Capabilities

- Defined in section 5.1.2.

##### Section 6.2.3 [applicable]: Header Format

- The unmodified TCP header format applies, with the SCPS options and related extensions specified in this document also applicable.

##### Section 6.2.4.1 [modified]: Interface Requirements

- The interface to the IP layer is defined in section 3.3.2.
- The interface to the applications is defined in section 3.3.1.

##### Section 6.2.4.2 [modified]: Connection Management and Data Transfer

- 1) **Push Flag** SHALL be supported as per SCPS-TP.
- 2) **Window management** SHALL be performed as per SCPS-TP.
- 3) **Urgent data** support is OPTIONAL.
- 4) **Initial sequence number selection** SHALL be as per section 5.1.1 above.

- 5) **Opening connections** SHALL follow the rules defined in SCPS-TP regarding the state machine.
- 6) **Closing connections** SHALL follow the rules defined in SCPS-TP regarding the state machine.
- 7) **Retransmissions** SHOULD be performed as defined in SCPS-TP. Modification or extensions (e.g., when not using congestion control or when using a different style of congestion control) are allowed as defined in this document.
- 8) **Generating acknowledgements** SHOULD follow the rules defined in SCPS-TP. Acknowledgment along the forward link SHOULD use standard acknowledgements. Acknowledgements to support acknowledgement sensitive congestion control schemes for the return link. Acknowledgements along the return link MAY use Acknowledgement Frequency Reduction.
- 9) **Sending data** SHALL follow the rules defined in SCPS-TP with the exception of record boundary marking.
- 10) **Connection failures** SHALL be treated as per SCPS-TP.
- 11) **Sending Keep-Alive Packets** SHALL follow the rules defined in SCPS-TP. Keep-alive packets may optionally be sent more frequently by an I-PEP.

Section 6.3 [not applicable]: UDP Requirements

- Not applicable.

Section 6.4 [not applicable]: Network Management Requirements

- Not applicable.

## 5.1.2 SPECIFIC IETF RFCS REFERENCED BY SCPS-TP

This subsection addresses the extensions to plain TCP (RFC 793) and the general requirements on Internet hosts (RFC 1122) that are referenced by SCPS-TP and discusses how the respective functions are used by the I-PEP transport specification.

This subsection makes explicit reference to SCPS-TP section 6.2.2 (Mission-specific Requirements).

The following rules apply when adopting this feature set of SCPS-TP for the I-PEP transport.

Section 6.2.2.2 [applicable]: Window scaling option

Window scaling SHOULD be supported following RFC 1323 (see also next item).

Section 6.2.2.3 [modified]: Timestamp option

Support for TCP timestamps as defined in RFC 1323 is optional. TCP Timestamps are not used unless TCP is used as the I-PEP congestion control mechanism. Given the round trip time variability of DVB-RCS they can be of limited use. TCP Timestamps are not useful for rate control and TCP Vegas congestion control, as they do not use timestamps to operate, and other congestion control algorithms may also not use timestamps. I-PEP implementations SHOULD be able to use only the window scaling part of RFC 1323 but they MAY support TCP timestamps as defined in RFC1323 as an option.

Section 6.2.2.4 [modified]: TCP for Transactions (T/TCP)

RFC 1644 (TCP for Transactions) is optional for I-PEP implementations. The use of T/TCP can add value to satellite communications because it eliminates the extra round-trip for TCP connection setup. It also provides for an accelerated connection tear down, which permits resource minimization for I-PEP entities.

It should be noted that T/TCP is vulnerable to denial-of-service (DoS) attacks from within the satellite network: a client faking a source IP address and port can cause the peer server to perform operations involving computation and memory resources and direct the response of the server to an unsuspecting target, consuming bandwidth in general and resources on the target. Note that as of January 2005, the IETF RFC 1644 is considered as a historic extension.

Section 6.2.2.5 [applicable]: Selective Negative Acknowledgement (SNACK)

The use of SNACK (as defined above) **MUST** be negotiated using the SCPS-TP Capabilities Option field.

Selective Acknowledgements (SACK) as described by RFC 2018 and RFC 2883, are optional for implementation. While SNACK is the preferred enhanced acknowledgment scheme, SACK may be useful in environments with partial deployment of I-PEPs thus being compatible with end-hosts using SACK.

Section 6.2.2.6 [not applicable]: Record Boundaries option

This **MUST NOT** be used. It is considered to be the respective application's responsibility to provide for proper record marking (if needed).

Section 6.2.2.7 [modified]: Header Compression

See section 5.1.3.

Section 6.2.2.8: Acknowledgement Frequency Reduction (AFR)

AFR can add considerable benefits for return channel bandwidth reduction. However, implementing this against an ACK-based sender can result in poor performance. Please see section 5.1.4 on I-PEP Congestion Control for detailed considerations.

Section 6.2.2.9 [not applicable]

Section 6.2.2.10 [modified]: Non-use of congestion control

Refer to section 5.1.4 for the specification of congestion control.

Section 6.2.2.11 [not applicable]

Section 6.2.2.12 [not applicable]

### 5.1.3 SCPS-TP HEADER COMPRESSION AND CLARIFICATIONS

SCPS-TP Header Compression is **OPTIONAL** for implementation. The impact of header compression in the forward channel is minimal; however, the return channel bandwidth saving for acknowledgements can be an important advantage.

Header compression **MUST** be negotiated using the SCPS-TP Capabilities Option field.

Some ambiguities in the SCPS-TP specification have been identified since its original publication. These include the clarification on the use of compressed SNACK options and the addition of ECN support. Clarifications have been put forward as follows.

Table 3-2 of SCPS-TP should read:

**Table 3-2 (of SCPS-TP): Compressed Header Bit-Vector Contents**

Bit Name	Meaning when set to '1'	Notes
More	Compressed Header Bit-Vector is 16 bits long rather than 8 bits long.	
TS1	TCP Timestamp Option is present.	See 6.2.2.5
TS2	A timestamp reply (TS Echo Reply) appears in the compressed header.	See 6.2.2.5
RB	The last octet of data accompanying this segment is the end of a user-defined record.	See 3.3.1
Snack	The compressed header contains a short-form SNACK option.	
P	The Push bit from the uncompressed TCP header is set to '1'.	
S	The compressed header contains a 4-octet sequence number.	
A	The compressed header contains a 2-octet window specification and a 4-octet acknowledgment number.	
Opts	The compressed header contains uncompressed options.	
Pad	The compressed header contains one octet of padding.	
URG	The URG bit from the uncompressed TCP header is set to '1'.	
AckR	The ACK bit from the uncompressed TCP header is set to '1' (this field is valid only when the RST bit is set to '1').	
ECE	The ECN-Echo flag from the uncompressed TCP header is set to '1'.	RFC 3168
RST	The RST bit from the uncompressed TCP header is set to '1'.	
CWR	The ECN congestion window reduced flag from the uncompressed TCP header is set to '1'.	RFC 3168
FIN	The FIN bit from the uncompressed TCP header is set to '1'.	

And the compressed SCPS-TP header should read as:

0	1	2	3	4	5	6	7	
<b>Connection ID</b>								Octet 1
<b>More</b>	<b>TS1</b>	<b>TS2</b>	<b>RB</b>	<b>Snack</b>	<b>Push</b>	<b>S</b>	<b>A</b>	Octet 2
<b>Opts</b>	<b>Pad</b>	<b>URG</b>	<b>AckR</b>	<b>ECE</b>	<b>RST</b>	<b>CWR</b>	<b>FIN</b>	Octet 3
<b>URG: Urgent Pointer (2 octets)</b>								Octet 4
<b>A: Window (2 octets)</b>								.
<b>A: Ack Number (4 octets)</b>								.
<b>S: Sequence Number (4 octets)</b>								.
<b>SN: Short-Form SNACK (4 octets)</b>								
<b>TS1: Outbound timestamp (format-dependent)</b>								
<b>TS2: Echo reply timestamp (format-dependent)</b>								
<b>Opts: Uncompressed TCP Options Length (1 octet)</b>								
<b>Opts: Uncompressed TCP Options (data-dependent)</b>								
<b>Pad: Optional Pad (1 octet)</b>								
<b>Checksum Octet 1</b>								
<b>Checksum Octet 2</b>								
<b>Data</b>								Octet $9 \leq n \leq 64$

**Figure 3-5 (of SCPS-TP): Compressed SCPS-TP Header**

The compressed short-form SNACK bit SHALL be set whenever a compressed short-form (i.e. NOT including a SNACK bit vector) SNACK option is present. The compressed short-form SNACK option SHALL occupy the 4 octets immediately following the sequence number field. The first two bytes of the compressed short-form SNACK option SHALL contain the *hole1 offset*. The next two bytes SHALL contain the *hole1 size*.

NOTE – Other SNACK options, including other short-form and all long-form options (those with uncompressed option lengths greater than 6 bytes) must be carried in the UNCOMPRESSED TCP Options portion of the compressed header.

When using SCPS-TP header compression, the ‘Connection ID’ for constructing SCPS-TP headers should be the Connection ID that was provided by the peer during SYN exchange.

The protocol id to be placed into the IP header when using header compression is 105. It has been agreed that this should be the protocol number used when computing checksums. The implementation of header compression causes a switch in protocol numbers to avoid confusion with TCP; as a result there is a loss of TCP header transparency. For communications between two I-PEP peers, this is not an issue.

#### 5.1.4 I-PEP CONGESTION CONTROL

SCPS-TP Section 3.4.2, which addresses congestion control, **SHOULD** be considered modified. Specifically, Congestion control schemes other than TCP, Rate Control or TCP Vegas **MAY** be used.

The congestion control scheme **SHOULD** be adaptive in both the forward and the return link, either through a standard or proprietary bandwidth estimation scheme, or direct coordination with the equipment.

Note that some congestion control schemes rely on ACK clocking to increase their transmission rate. For such schemes, the use of Acknowledgement Frequency Reduction (AFR) may result in poor performance. Therefore, AFR **SHOULD NOT** be used if the sender relies solely on counting ACKs (as opposed to counting bytes or otherwise estimating the transmission rate) to increase its congestion window.

SCPS-TP Section 6.2.2.10, which permits the non-use of congestion control at all under well-defined circumstances (and with the transmission path being under control by the sending I-PEP entity), applies with the preconditions listed there.

Support for ICMP Source Quench (as per section 6.2.4.1) is **OPTIONAL**. It is **NOT RECOMMENDED** as a primary means to relay congestion notification between inter-network nodes, in favour of ECN (see section 5.1.5).

When explicit congestion notification (RFC 3168) is implemented, an explicit congestion notification via the receipt of a segment with the ECN-Echo (ECE) bit set must be treated as an indication of network congestion. In this case, receipt of a segment with the ECE bit set **MUST** invoke the same congestion response as a lost segment.

### 5.1.5 EXPLICIT CONGESTION NOTIFICATION

Explicit Congestion Notification (ECN) as described by RFC 3168 is RECOMMENDED for implementation. It was developed following the publication of the SCPS-TP standard so there is currently no mention of ECN in the existing SCPS-TP standard.

Implementations that MAY operate in ECN-capable networks SHOULD support ECN functionality as described in RFC 3168. Implementations supporting ECN:

- a) MUST negotiate ECN capability during the SYN exchange or as part of the I-PEP session context;
- b) SHOULD signal to the network their ability and willingness to respond to ECN signals;
- c) MUST invoke congestion control in response to ECN signals in essentially the same way as to dropped packets.

ECN MAY be compressed using the SCPS-TP compression bit vector. See 5.1.3 SCPS-TP Header Compression and Clarifications for further information.

It should also be noted that DoS attacks are possible with ECN. I-PEP peers SHOULD therefore only react to ECN if they can assume that the packet originated from or is echoed by a “trusted source”.

## 5.2 *Transport Protocol Extensions*

### 5.2.1 INTRODUCTION TO EXTENSIBILITY MECHANISMS

This section presents standardized mechanisms for the transport layer per-connection extensibility.

The SCPS-TP Extension Option is first presented, followed a description by the I-PEP specific extended assignment. An alternative to the SCPS-TP Extended Option, called the TCP Option Data, is then presented in 5.2.4.

Using either of these extensibility mechanisms, specific uses of the I-PEP option are presented. These are generally targeted at integration with a higher session layer between I-PEP nodes.

The SCPS-TP Extended Option is an attractive solution for extensibility; however, it is limited by existing TCP Option limitation of a total of 40 bytes. For negotiation of options requiring larger options, the TCP Option Data solution can be used. Both of these solutions can be used interchangeably although the SCPS-TP Extended Option within TCP Option Space is recommended where possible. The specification addresses only options negotiated during the SYN connection setup phase. The use of extended options following this phase is determined on an Option specific basis. This specification does not include any use of Extended Options following the SYN phase of TCP/SCPS-TP connection setup.

### 5.2.2 SCPS-TP CAPABILITIES OPTION EXTENSION

The SCPS-TP Extended Capabilities Option is a mechanism to extend the per-connection negotiation for specific handling (reference to a session ‘connection class’, indicating a target address, etc.) or additional tagging (e.g. special QoS) or other evolutions of SCPS-TP options. This option permits the negotiation of other options beyond the SCPS Capabilities Option. The SCPS Extended Option is defined such that it permits an open path along which the SCPS-TP standard can be extended as requirements for transport-layer per-connection as the protocol evolves.

The following format is used to signal ‘extended’ SCPS capabilities. Extended capabilities allow endpoints to perform signaling in addition to that supported by the ‘standard’ SCPS Capabilities option described above.

Extended capabilities are identified by reuse of the SCPS Capabilities Option (option 20) two or more times on a particular SYN packet (TCP packet with the SYN bit set). These extended (20) options are prohibited from having length 4 to help differentiate them from ‘standard’ SCPS Capabilities Options. The FIRST SCPS Capabilities Option present on a SYN segment MUST have length 4 and is interpreted as above.

### 5.2.2.1 Extended Capability Format

The extended capability is indicated by the presence of a second (third, etc.) SCPS Capabilities Option (TCP option 20). This option has the standard TCP option format, as shown in Figure 3-2 of SCPS-TP. The type of this option is SCPS Capabilities (20), and the length is variable, but **MUST** be larger than 4 (the mandated length of the base SCPS Capabilities option above) as depicted in figure 9.

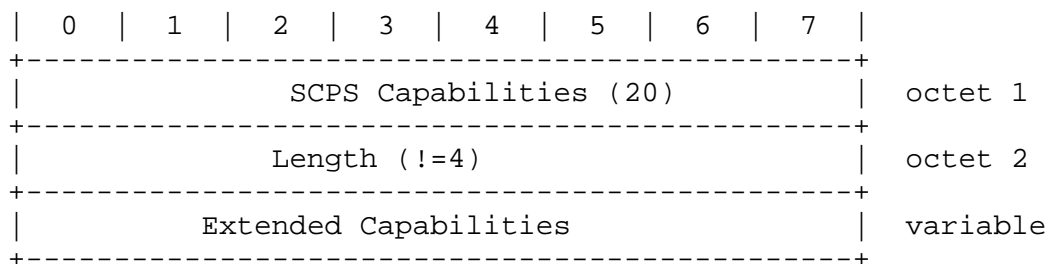


Figure 8: Beginning of extended capabilities signaling

Each 'Extended Capability Option' consists of:

- 8 bits defining an 'extended option binding space';
- 4 bits defining the length of the option binding space (in 16-bit words);
- 4 bits usable as flags, or some other binding space specific definition.

Figure 9 below shows the format for each individual extended capability.

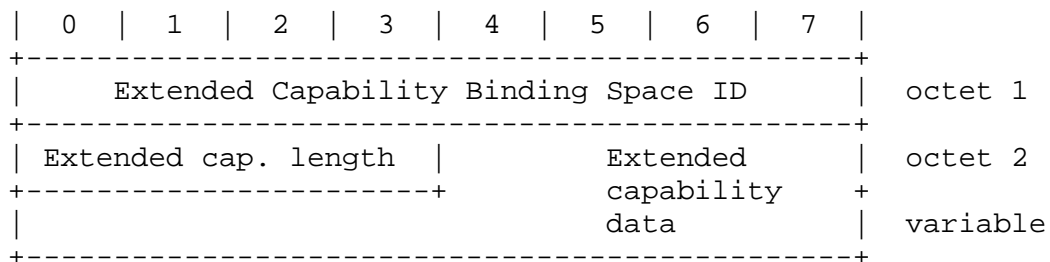


Figure 9: Format for extended capabilities

### 5.2.2.2 *Extended Capability Fields*

#### **Extended Capability Binding Space ID:**

This identifies the particular extended capability binding space. The extended capability data is interpreted in the context of this identifier.

#### **Extended Capability Length:**

The extended capability length field specifies the length of the extended capability, exclusive of the octets containing the SCPS-TP Extended Capability Type (20) and TCP Option length, in 16-bit words. Thus, in Figure 3-3 of SPCS-TP, an extended capability length of 1 would indicate that only octets 1 and 2 in the figure were present.

NOTE – The minimum length of an extended capability is 2 octets (the octet containing the binding space identifier and the octet whose first 4 bits contain the length).

The length of the Extended Capability Option plus all preceding and succeeding options **MUST** fit into the TCP option field unless the TCP option space is extended as described in section 5.2.4 below. In this latter case, all IP and TCP headers and options together **SHOULD** be smaller in size than the path MTU. If the path MTU is unknown a value of 1500 bytes **SHOULD** be assumed.



0	1	2	3	4	5	6	7	
SCPS Capabilities (20)								octet 1
Length (!=4)								octet 2
Extended Capability Binding Space ID1								octet 3
Extended cap. len1				Extended capability				octet 4
data1								variable
SCPS Capabilities (20)								octet n
Extended Capability Binding Space ID2								octet n+1
Extended cap. len2				Extended capability				octet n+2
data2								variable

Figure 11: Using multiple SCPS Capabilities options to express multiple extended capabilities

#### 5.2.2.4 Extending the Extended Capability Binding Identifier Space

An extended capability binding space identifier value of 0xFF (decimal 255) is used to extend the extended capability binding space from 255 to 509 values. A value of 255 in the extended capabilities binding space identifier field indicates that the actual extended capability binding space identifier is calculated by adding 256 to the value of the octet following the octet containing the extended capability length (octet 5 in Figure 12 below). If octet 5 in Figure 12 was itself 0xFF (255), the extended capability binding space identifier would be 510 plus the value of the octet following octet 5.

NOTE – Using this extension method, the position of the extended capability length field is fixed relative to the start of the extended capability binding space identifier, regardless of the number of bytes needed to express the extended capability binding space identifier.

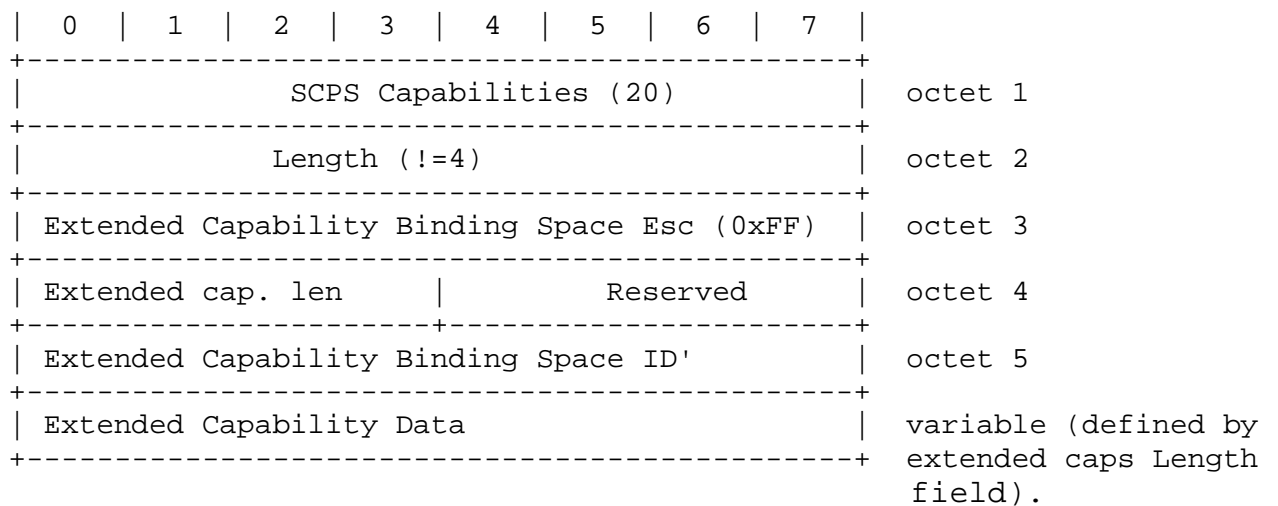


Figure 12: An extended SCPS-TP capability specified by a binding space identifier in the 250-509 range.

If extended capability binding space extensions are in use (i.e. octet 3 of figure 12 is 0xFF) then the 4 bits following the extended capability length are set to zero on transmission and **MUST** be ignored on receipt.

### 5.2.2.5 Meanings of Specific Extended Capability Binding Space Identifiers

#### 5.2.2.5.1 Processing Unrecognized Extended Capability Binding Spaces

As with TCP options, implementations **MUST** ignore extended capability binding spaces that they do not understand. In these cases the implementations can read the extended capability length field and skip over the unknown data to continue processing the rest of the extended capabilities (if any).

#### 5.2.2.5.2 Standard Extended Capabilities Binding Space Identifiers

Values 0x00 through 0x0F for the extended capability binding space identifier are reserved for standards use.

#### 5.2.2.5.3 Experimental Extended Capabilities Binding Space ID

Extended capability binding space identifier 0xFE (254) is reserved for experimentation. Implementations using this experimental extended capability binding space **MUST** take precautions to ensure that they are interpreting the extended capability data correctly. I-PEP implementers **SHOULD** instead use their Vendor ID where possible.

### 5.2.3 THE I-PEP EXTENDED OPTION

The extended I-PEP option uses the binding space of ‘100’ in decimal, or ‘0x64’ in hexadecimal.

The following options are defined in this specification.

Option	Value	Length	Description
TCP Option Data	0x5	4	Indicates that additional space in the TCP payload of the SYN packet is allocated to carry further TCP options.
Session ID and Service Tag	0x1	6	Links an incoming TCP connection to an existing session context; the Service Tag supports future application-specific enhancements and allows requesting invocation of such an enhancement function on a per-connection basis
Target Address	0xA	8	Used to explicitly signal the target IP address of the ultimate destination to the I-PEP peer. This allows for explicit routing to a peer using the destination IP address and port number without requiring IP-in-IP encapsulation.
Vendor ID	0x8	$\geq 4$	Used to convey the vendor identification of the I-PEP peer and thus provides a first mechanism for the receiving I-PEP peer to determine the available feature set.
Option Space Extension	0xF	$\geq 4$	This value is reserved for future extension of I-PEP options. If the I-PEP Extended Option has this value the immediately following octet contains is added to the option identifier. I-PEP options in the range of 0 – 14 (0x00 – 0x0E) are hence encoded in the four option bits, I-PEP options in the range of 15 – 270 (0x0F – 0x10E) are encoded by the sum of the option space extension and the succeeding octet.

**Table 1: I-PEP Extended Options**

The length represents the total length in octets of the option, not including the introductory SCPS-TP Capability and Length TCP Option header. Note that the length field of the Extended SCPS-TP Capabilities Options is described by section 5.2.2.2.

## 5.2.4 TCP OPTION DATA NOTIFICATION

The option space provided by TCP is fairly limited which impedes elaborate negotiation of options on a per I-PEP transport connection basis and may prevent communicating parameter sets upon connection establishment. To address these shortcomings, the TCP Option Data option MAY be used in the TCP packet containing the SYN bit to provide additional option space prior to the first byte of application data. The TCP Option Data Option is an I-PEP Option.

The TCP option data field MUST be the last option in the regular TCP option sequence (which MUST also be reflected in the TCP option length field). Immediately afterwards, MUST follow other options in the TCP data.

If present in the TCP option field, the TCP Option Data option indicates that additional option data is present in the beginning of the TCP segment payload *prior* to any application data (since, unless T/TCP is used, the data segment is unused anyway, including additional options is not an issue). The option data MUST NOT consume TCP sequence number space and MUST be implicitly acknowledged with the SYN flag. That is, regardless whether (and irrespective of how many bytes) or not TCP Option Data are included in a SYN packet carrying the initial TCP sequence number *seq*, the corresponding ACK packet MUST acknowledge receipt of the entire SYN packet (including possible TCP Option Data) with a TCP acknowledgement number of *seq+1*. The first data segment transmitted in the direction of the SYN packet MUST then bear the sequence number *seq+1*.

The second half of octet 2 will be used to describe that the I-PEP sub-option type “TCP Option Data” as ‘0x5’, represented in hexadecimal. The option format is depicted in figure 13.

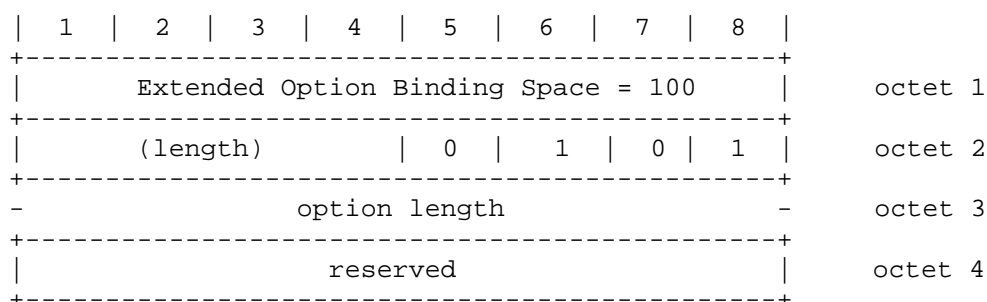


Figure 13: Format of the TCP Option Data sub-option

The fields have the following semantics:

**Option Length:**

The *option length* field indicates how many 16 bit words of option data are included *in the TCP data portion following the TCP option field*.

It **MUST** be the *last* sub-option and be included in the last regular TCP option contained in the TCP option space (and the TCP option length field **MUST** be set accordingly). Those TCP options located in the regular TCP option space and those located in the TCP data part of the SYN segment **MUST** be treated jointly as a single set of options by the receiver.

The total TCP SYN segment (including all data options) **MUST** fit into a single IP packet and **SHOULD** be less than the path MTU size.

TCP Option Data **MAY** carry arbitrary TCP options including the Extended Capability Option as defined above. Thus, multiple options can be specified within the TCP Option Data. Padding to a multiple of 16 bit words **SHOULD** follow the End-of-Option-List or No-Operation option as defined in RFC 793.



0x00: A Session ID of 0x00 indicates that no association with an I-PEP session context is intended. (This allows only the Service Tag to be used.)

## 5.2.6 TARGET ADDRESS

The Target Address option **MAY** be used in a SYN packet to explicitly indicate the ultimate destination to the I-PEP peer entity, e.g., when the connection forwarding is non-transparent. This option **MUST** only be used when previously negotiated at the session layer or otherwise known (e.g., statically configured). This option **MUST NOT** be used in any other packet than a SYN packet.

The second half of octet 2 (at the start of the sub-option specific data) will be used to describe that the I-PEP sub-option type “Target Address” as ‘0x0A’, represented in hexadecimal.

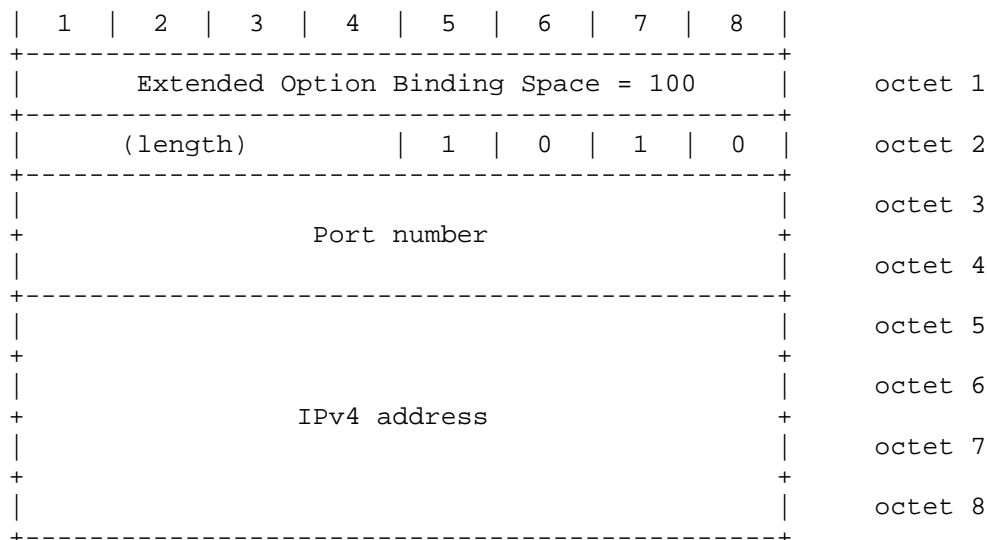


Figure 15: Target address option

The fields have the following semantics:

**reserved:**

This field **MUST** be set to zero upon transmission and **MUST** be ignored upon reception.

**Port number:**

This field contains the port number of the remote application peer.

**IPv4 address:**

This field contains the IPv4 address of the remote application peer.

### 5.2.7 VENDOR ID

The I-PEP Vendor ID can be used by I-PEP implementers for private extension of the protocol for other value added services.

The assignment of Vendor IDs will be managed by SatLabs. I-PEP Implementers wishing to be assigned a Vendor ID should do so via the SatLabs web site ([www.satlabs.org](http://www.satlabs.org)).

The Vendor ID option MAY be used in a SYN packet to indicate the manufacturer of the originating I-PEP peer, the product code, and the major and minor version numbers. This option SHOULD NOT be used in other than SYN segments.

Octet 4 (at the start of the sub-option specific data) will be used to describe that the I-PEP sub-option type “I-PEP Vendor ID” as ‘0x8’, represented in hexadecimal.

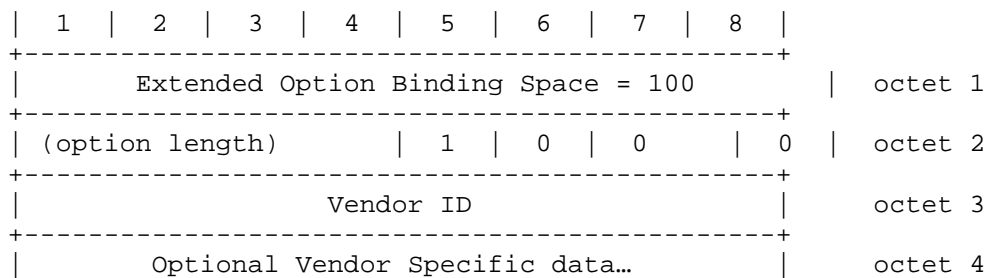


Figure 16: Vendor ID Option

The field has the following semantics:

•  
**Vendor ID:**

An 8 bit value assigned to the vendor upon request (related to the extended option binding space identifier). The value of 0xFF is reserved for future extension (following 255 assignments) such that two or more octets may make up this value. For such extensions, the same rules as for the extended option binding space apply.

Vendors are highly encouraged to include their own versioning fields as vendors will only be assigned one Vendor ID each.

## 5.2.8 I-PEP OPTION SPACE EXTENSION

The I-PEP option space can be extended to also make use of the octet immediately following the option identification. In this case, the numerical values of the suboption types and the following option are numerically added thus yielding a total suboption space of 0 – 270 (0x00 – 0x10E).

This is indicated by the I-PEP suboption type '0xF'. The resulting packet format is depicted in figure 17.

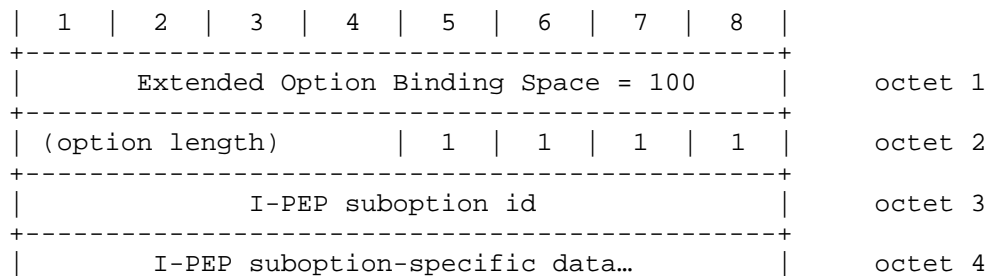


Figure 17: I-PEP Option Space Extension

The field has the following semantics:

**I-PEP suboption id:**

An 8 bit value to be added to 15 (0x0F) to determine actual I-PEP suboption identifier.

**Suboption-specific data:**

Data as defined by the respective I-PEP suboption.

## 6 SESSION LAYER FRAMEWORK

The previous section has described a set of extension to TCP based upon the space communications standard SCPS-TP resulting in the I-PEP transport protocol specification. A particular feature of the I-PEP transport protocol is to provide enhanced performance via long-delay links (such as satellites) and allow for broader flexibility in algorithm selection—so that I-PEP peers can be adapted better to the conditions of the satellite communication environment they are operating in. The I-PEP transport specification allows these options to be negotiated on a per-connection basis, using the TCP option fields to carry the respective parameters.

In many satellite communication setups, two I-PEP peers are likely to share a number of parallel I-PEP transport connections. In such a case, it is desirable to allow peers to negotiate a common set of parameters for all those I-PEP transport connections as well as to establish parameters beyond those negotiable in the—fairly restricted—set of TCP options.

For this reason, a session layer framework is provided. In addition to exchanging data via individual transport connections, two I-PEP peers may set up an *I-PEP session* and thus create a shared context from which the parameters for future I-PEP transport connections can be derived.

Support for an I-PEP session (and hence the associated session protocol) is **OPTIONAL**.

The I-PEP session layer may be used to provide the following functions (each of which is again optional even when the session layer is implemented):

- server location and service identification;
- session setup (and later teardown) and establishment of a shared context for transport connections and other functions;
- feature negotiation between peers;
- client (and server) authentication;
- establishment of a shared secret; and
- vendor-specific extensions.

This section provides a concise overview of the I-PEP session layer and shows how I-PEP transport and I-PEP session layer are integrated in the I-PEP session framework.

The I-PEP session layer protocol is specified in a companion document.

## 6.1 Associating I-PEP Sessions and Transport Connections

As mentioned above, two I-PEP peers may decide to establish a shared session context in which they may negotiate feature sets to be used with future I-PEP transport connections between those two peers. When I-PEP transport connections are established, they may—but need not—be associated with an existing session context.

Figure 18 depicts the general concept of I-PEP transport connections and an I-PEP session: Two I-PEP peers share a number of I-PEP transport connections and, at some point, an I-PEP session is set up. Some subsequently established transport connections make reference to this session context.

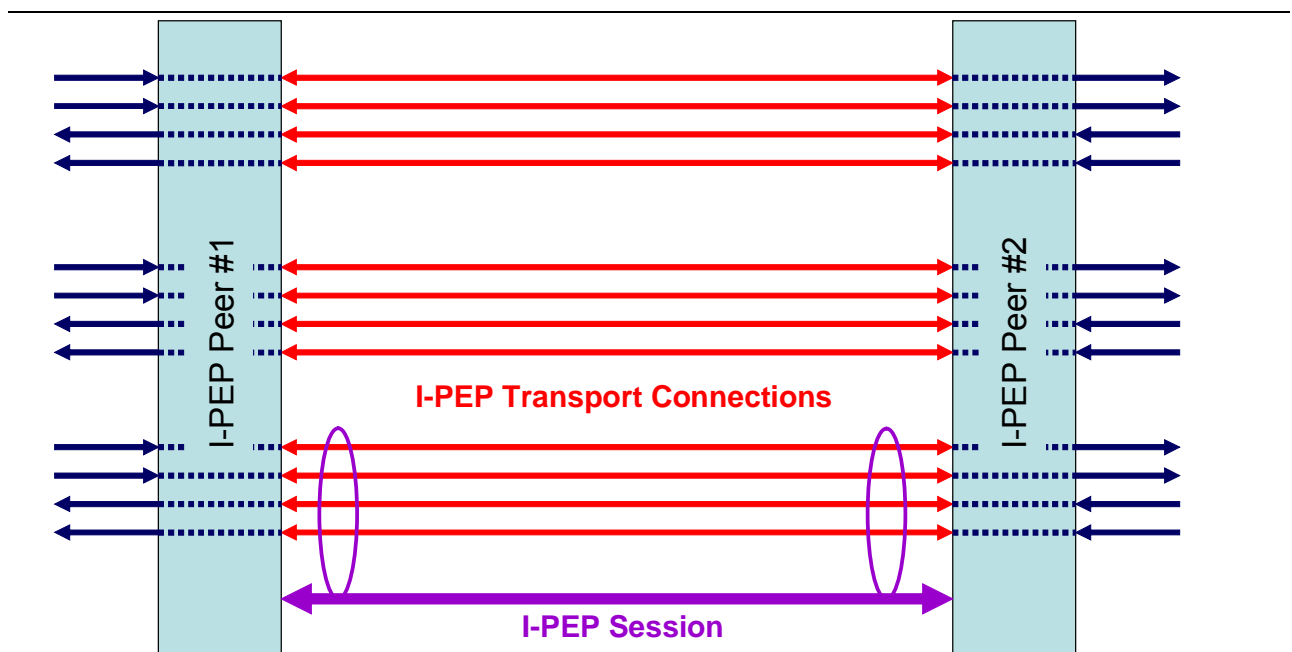


Figure 18: I-PEP clients accessing multiple I-PEP servers

The integration of I-PEP transport connections and I-PEP sessions works as follows: an I-PEP session context is identified by means of its I-PEP peer transport addresses (IP address, port number, and further identification information) and a session context identifier (*Session ID*) chosen between the two peers. All parameters negotiated within this session context make reference to the Session ID so that, in principle, even multiple sessions could exist between two peers.

When a new I-PEP transport connection is set up a specific TCP options field (Session ID) may be used to cross-reference the session context, thereby binding the I-PEP transport connection to the session. If no Session ID is specified no binding is established. Within a session, additional services may be provided which an I-PEP transport connection may reference by means of a Service Tag. Details of such services are provided in the I-PEP session layer specification.

## 6.2 Protocol Outline

The I-PEP session protocol runs on top of TCP (or I-PEP transport) and thereby inherits reliability and congestion control mechanisms from its underlying transport. A UDP-based mechanism is supported for multicast service announcements; this transport mechanism uses rate control and repeated transmissions peered with soft state for reliability.

The following features are supported by the I-PEP session protocol:

- Announcing servers and server capabilities

The I-PEP session protocol allows a server to announce its availability (and the availability of redundant/load balancing other servers). The announcements may include the vendor and software version, the service provider (and associated parameters such as tariff) as well as basic capabilities of the I-PEP server. Vendor-specific extensions are supported as well. The announcement occurs in configurable intervals. The server may also announce that it will be taken out of service shortly.

This information allows an I-PEP client to determine which services are available (possibly offered by different I-PEP servers and service providers) and pick the right I-PEP server to connect to.

The same announcement can be targeted via unicast (i.e., via an existing TCP connection) at a client to update this particular client's state information about the server.

Announcements are soft-state and are never confirmed—in order to allow for a uniform processing of unicast and multicast announcements on the client side.

- Session Setup

A two-way handshake is provided for simple session setup. A session can be set up at any time and its parameters may take effect for future I-PEP transport connections (if so referenced when setting up an I-PEP transport connection). The information about the I-PEP server to connect to (and its TCP transport address) may have been received in a previous announcement, obtained out of band, or configured manually.

During the session setup, a message may carry independent parameter sets for the following functions:

- one-way or mutual identification and authentication;
- capability description and negotiation;
- Diffie-Hellman keying information to establish a shared secret; and
- additional parameters as part of the session context.

In particular, these parameters may be used to determine whether or not the server is capable of supporting application-specific functions such as HTTP prefetching.

Furthermore, vendor-specific extensions may be included.

One parameter communicated during session setup is an identifier for the respective session. This **Session ID** is also used to link an I-PEP transport connection to the respective session and allow the transport connection to be initialized from the pre-established session context. To link a particular I-PEP transport connection to an existing session context, the SYN packet for the I-PEP transport connection must include the “Session ID and Service Tag” option in the TCP header—which is used to provide the necessary binding.

In addition, a set of **Service Tags** may be announced along with an I-PEP session context. These service tags may be used to identify application-layer services for an I-PEP transport connection and provide the basic hook to enable later integration of features such as HTTP prefetching. A TCP SYN packet (or the initial segment) may include such a Service Tag and thereby indicate that additional processing is desired for the respective transport connection.

- Session Update

Session context may also be updated from either I-PEP peer during a session.

- Session Teardown

An I-PEP session may be torn down by either I-PEP peer. With the confirmation of the teardown, the session context is deleted.

It is up to each implementation to decide what to do with still active I-PEP transport connections previously associated with the session just deleted. If I-PEP sessions are used for authentication and accounting, it is recommended that these I-PEP transport connections are released immediately. Otherwise, they may continue to exist (for some predefined time).

Subsequently established I-PEP transport connections can no longer refer to the session context but need to negotiate all of their parameters on their own. In any case, the establishment of new I-PEP transport connections referencing the deleted session should be refused.

## 7 CONFORMANCE SPECIFICATION

This section discusses general interoperability aspects of the I-PEP specification. Section 7.1 introduces the minimal functional requirements, section 7.2 describes test tools, section 7.3 describes a number of interoperability scenarios, and, finally, section 7.4 defines the PICS usage and provides a template for a conformance statement.

### *7.1 Minimal Functional Requirements and Tests*

The PICS proforma describes the minimum capabilities required of an I-PEP compliant entity.

Testing of compliance can draw heavily from available technologies for functional testing, including the following:

2. As TCP compatibility is required, much of the core TCP state machine can be tested by using existing operating systems as peer test systems. Testing against open source operating systems such as Linux or BSD may provide additional flexibility for test and analysis.
3. The SCPS-TP Reference Implementation is freely available and can be used to test key I-PEP capabilities, including:
  - a) The SCPS-TP Capability option
  - b) SNACK
  - c) SCPS-TP Header Compression
  - d) Rate Control and Vegas congestion control
  - e) Demonstrate behaviour of a PEP/gateway

[http://www.scps.org/Reference\\_S\\_W/reference\\_s\\_w.html](http://www.scps.org/Reference_S_W/reference_s_w.html)

3. Additional I-PEP capabilities such as RFC 3168, RFC 1323, and RFC 1644 are also available in a number of open source operating systems.

While the I-PEP specification is a profile with modifications and extensions to TCP and SCPS-TP, validating the core functions of an implementation can be done using the mechanisms described above.

Other capabilities currently unique to the I-PEP transport specification including the Extended SCPS-TP Capability Option can be tested through interoperability tests with other I-PEP implementations. In face of a full I-PEP implementation, these represent reasonably manageable and clear capabilities suitable for simple and brief interoperability testing.

## 7.2 *Test Tools*

Many tools currently exist for the test, analysis of TCP/IP, for example, tcpdump and Ethereal. SCPS-TP yields an advantage as it can be further analysed with tools such as 'tcptrace' and other both open source and commercial tools. The SCPS-TP capability option will simply show as an unrecognized TCP option. SCPS-TP Header Compression loses TCP header transparency, however with a complete tcpdump capture there exist tools to convert a complete header-compressed session to conventional TCP. One of these tools is also made freely available with the Reference Implementation of SCPS-TP.

### References

- Test and Analysis Tools

<http://www.tcpdump.org/>  
<http://jarok.cs.ohiou.edu/software/tcptrace/tcptrace.html>  
<http://www.scps.org/>  
<http://www.ethereal.com/>  
<http://www.web-polygraph.org/>  
<http://www.pcausa.com/Utilities/pcattcp.htm>

- Satellite Link Emulation

Note that these software tools are useful in simulating a raw satellite links, but not DVB-RCS networking whereby capacity may be assigned many different ways leading to considerable variability in both bandwidth availability as well as round trip time.

[http://info.iet.unipi.it/~luigi/ip\\_dummysnet/](http://info.iet.unipi.it/~luigi/ip_dummysnet/)  
<http://www-x.antd.nist.gov/nistnet/>

### ***7.3 Interoperability scenarios***

This section describes several core scenarios that need to be workable with an I-PEP implementation. For the scenarios, we assume that I-PEP client and server implementations from four different vendors A, B, C, and D are available and that these may be available in different revisions, e.g., A1 and A2. The following list is not exhaustive but is rather meant to characterize the classes of interoperability scenarios:

#### **7.3.1 SCENARIO 1: I-PEP CLIENT A AND I-PEP SERVER A**

If two I-PEP peers from the same vendor communicate, they are expected to be able to identify each other and negotiate the full feature set available from the respective vendor. After setup, arbitrary vendor-specific extensions may be invoked.

#### **7.3.2 SCENARIO 2: I-PEP CLIENT A1 AND I-PEP SERVER A2**

Two I-PEP peers from the same vendor running different versions of the I-PEP transport and session protocol and/or vendor-specific protocol (extensions) are able to agree on a common denominator to use for communications. They will be able to mutually determine their respective capability set and then use all the functionality commonly available to them.

A more advanced server will restrict its features to the capabilities of a less capable client and vice versa.

#### **7.3.3 SCENARIO 3: I-PEP CLIENT A AND I-PEP SERVER B**

An I-PEP client and a server from different vendors will be able to communicate based upon the I-PEP specification. They will be able to locate and contact one another and they will be able to negotiate the standardized options defined in (future versions of) of this specification. The two entities will also be able to determine vendor-specific options they may have in common and thus will be able to utilize the maximum common feature set supported.

Two I-PEP peers from different vendors will be able to interoperate at least using the mandatory features defined in version 1.0 of this specification.

Future versions of this specification will be defined in a backward-compatible fashion so that a version 5 client can talk to a version 1 server and vice versa. The mandatory parts of version 1.0 of this specification will provide the basis for all future I-PEP versions.

#### 7.3.4 SCENARIO 4: I-PEP CLIENT B AND I-PEP SERVER A

The considerations of scenario 3 apply. It is irrelevant which vendor provides the I-PEP server and which provides the client.

#### 7.3.5 SCENARIO 5: I-PEP CLIENTS A, B, C1, AND C2 AND I-PEP SERVER C2

In a heterogeneous environment, different I-PEP clients may operate in the same DVB-RCS network. In such a case, each client will negotiate the usable feature set individually with the I-PEP server. If client and server are from different vendors (clients A and B), they will negotiate a common feature set based upon this specification and will at least be able to interoperate at the mandatory functions supported by version 1.0 of this specification (see scenarios 3 and 4). If client and server are from the same vendor (clients C1 and C2), scenarios 1 and 2 apply.

It will be up to the server implementation to ensure that all connections to different clients are treated independently and that, at the same time, e.g. different approaches to congestion control supported by different clients do not lead to unfair link utilization among the different connections.

#### 7.3.6 SCENARIO 6: I-PEP CLIENTS A, B, C, AND D AND I-PEP SERVERS C AND D

In an even more heterogeneous environment, in addition to several I-PEP clients from different vendors (possibly running different versions of the respective products), several I-PEP servers may exist in parallel. In this scenario, each I-PEP client may pick / be assigned a different I-PEP server for its I-PEP transport connections.

In such a case, any same/different vendor combination of I-PEP client and server will be workable, at least at the level of the mandatory feature set supported by version 1.0 of this specification. A server may be assigned dynamically (e.g., by means of discovery, session layer negotiation, or explicit load balancing) to a client and vice versa. To improve average communication capabilities, servers and clients may be matched based upon their respective vendor to allow exploiting the maximum set of functionality for each connection. Furthermore, the various communication relationships do not interfere, regardless of the level of functionality supported.

With multiple I-PEP servers from the same and/or different vendors in operation, it is up to these servers to ensure fair utilization of the satellite link across all servers and clients (according to their respective service level agreements).

The same considerations apply to communications with an I-PEP session server (which may but need not be integrated with an I-PEP server).

For further considerations on the extensibility of the I-PEP specification, refer to section 9.

## 7.4 I-PEP PICS USAGE

### 7.4.1 INTRODUCTION

The I-PEP specification makes use of a Protocol Implementation Conformance Statement (PICS), in a similar but simplified fashion as the SCPS-TP specification. An I-PEP implementation must implement the mandatory capabilities specified to be considered compliant with the specification. The PICS statement provides a basis for SatLabs, implementers, and testers to verify compatibility with the I-PEP standard, to form a strong basis for interoperability, and can be used as a guide for additional enhancements as described the optional capabilities.

### 7.4.2 NOTATION

The following are used in the PRL to indicate the status of features:

#### Status Symbols

M	mandatory.
O	optional for implementation
NR	not recommended but permitted.

The support of every item as claimed by the implementer is stated by entering the appropriate answer (Y, N, or N/A) in the support column:

Y	Yes, supported by the implementation.
N	No, not supported by the implementation.
N/A	Not applicable.

### 7.4.3 I-PEP PICS STATEMENT

#### 1. Identification

Supplier	
Contact Point for Queries	
Implementation Name and Versions	
Other Information	

#### 2. Protocol Summary

I-PEP Protocol Version	
Vendor identifier code	

Date of Statement	
-------------------	--

#### 3. Major Capabilities

Item	Protocol Feature	Reference	Status	Support
3.1	TCP and reliable data transfer	RFC 793, 1122; I-PEP 3.2	M	
3.2	SCPS-TP Capabilities Option	SCPS-TP 3.2.3	M	
3.3	Extended Capabilities Option	I-PEP 5.2.2	O	
3.4	Short SNACK	SCPS-TP 6.2.2.5,	M	
3.4	SACK	RFC 2018	O	
3.5	SCPS-TP TCP Header Compression	SCPS-TP C2.5.2, updated by I-PEP 5.1	O	
3.6	RFC 1323: Window Scaling	RFC 1323	M	
3.7	RFC 1323: Timestamps	RFC 1323	O	

#### 4. Forwarding Functionality

Item	Protocol Feature	Reference	Status	Support
4.1	TCP Connection conversion to SCPS-TP/I-PEP	I-PEP 3.2.1, 8.1.2.1	M	
4.2	Transparent interception	I-PEP 3.2.1, 8.1.2.1	O	
4.3	Proxy-based operation	I-PEP 3.2.1, 8.1.2.1	O	
4.4	Passing UDP, IPSec, and other non-TCP IP packets outside the I-PEP transport (i.e. without performance enhancement)	I-PEP 3.2.1, 3.3.2, 8.1.2.1	M	
4.4	End-to-end connection setup semantics	I-PEP 3.2.1, 8.1.2.1	O	
4.5	Local connection accept	I-PEP 3.2.1, 8.1.2.1	O	

#### 5. Congestion Control Related Capabilities

Item	Protocol Feature	Reference	Status	Support
5.1	Forward Link Congestion Control	I-PEP 5.1.4	M	
5.1	Return Link Congestion Control	I-PEP 5.1.4	M	
5.2	Forward Link: Standard acknowledgements	I-PEP 5.1	M	
5.3	Return Link: Acknowledgement Frequency Reduction	SCPS-TP 6.2.2.8, I-PEP 5.1.4	O	
5.4	Explicit Congestion Notification (ECN)	RFC 3168, I-PEP 5.1.5	O	

## 6. Detailed I-PEP SCPS-TP Profile Minor Capabilities

Item	Protocol Feature	Reference	Status	Support
6.1	Long SNACK	SCPS-TP: 3.5.2.5	O	
6.2	ICMP Source Quench	SCPS-TP; RFC 792; IPEP 5.1.4	NR	
6.3	Best Effort Transport Service and Record Boundaries	SCPS TP: 3.3.2, 6.2.2.6, 6.2.2.11,	NR	
6.4	Random Initial Sequence Number Selection	I-PEP 5.1.1; RFC 793	M	
6.5	Corruption Experienced (CE) Option	SCPS-TP 3.4.2.5, 3.4.2.7.3	NR	
6.6	Multiple Forward Retransmission	SCPS-TP	NR	
6.7	I-PEP Transport connection context assignment	I-PEP 5.3.2.1	O	

## 8 SYSTEM ASPECTS

### 8.1 Gateways

In this section, we use the term *gateway* to refer to a specific I-PEP implementation functionally realized as a stand-alone device. The term *gateway* indicates that an I-PEP implementation provides transport (and application) layer functionality instead of just forwarding IP packets without considering their payload.

#### 8.1.1 TYPES OF GATEWAYS

I-PEP implementations may be used in a variety of settings as discussed in section 4. Depending on the operational setting, they may come in a variety of system types including (but not limited to):

- Symmetric gateways are used for plain peer-to-peer communication (“trunking”) between sites via a DVB-RCS hub, e.g., within an enterprise or between ISPs. In symmetric gateway scenarios, all gateways may be equally functionally capable.
- In contrast, asymmetric gateways exhibit—at least from an operational perspective (not necessarily functionally)—a clear separation into a server (=service provider) and client (service user) side. Such gateways are usually employed to offer Internet services via satellites to end users, enterprises, or service providers in remote locations.
- Gateways may operate transparently to the endpoints and user applications or they may require explicit configuration (statically or dynamically via protocols).
- Gateway may be generic, i.e. support enhancement of arbitrary applications or gateways may be limited to a subset of applications.
- Gateways may run in resource-constrained (CPU power, memory, etc.) embedded devices; they may run on powerful server machines; or (not in the focus of this specification though) they may be co-located (or even integrated) with the user devices and applications.
- Gateways may serve an arbitrary number of users: a single one, a small group, or several thousands.
- Depending on their application area, gateways may support an arbitrary number of applications and Application TCP connections in parallel: from less than ten to several tens or hundreds of thousands.

- Gateways may be able to operate a wide range of data rates in either direction: from a few kilobits per second to a hundred megabits per second and beyond.

### 8.1.2 GATEWAY FUNCTIONALITY

This section specifies the gateway functionality as experienced from *service user* point of view, in the sense that the gateway provides a performance enhancement service to the applications peers running on the end systems and thus ultimately to the users. This service interface may or may not be transparent to the application peers. This specification is deliberately constrained to be a high level description of user experience, as many detailed system aspects can safely be left to the implementations.

I-PEP entities usually operate peerwise (one running usually at or topologically close to either end of a DVB-RCS link). The resulting setup yields four interface points that are peerwise symmetric as depicted in figure 19 as external (“E”) and internal (“I”):

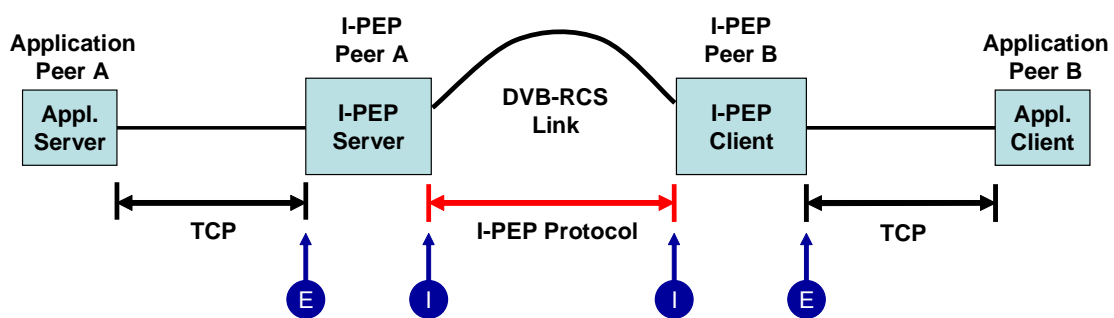


Figure 19: Reference interface points for I-PEP peers

The I-PEP (transport and session) protocol specification defines the *internal* interface (I) as referred to as the *air interface*, as done by sections 1 – 7 of this document for the I-PEP transport protocol.

The service behaviour relevant to application peers is defined per peer of I-PEP entities, i.e. *between* the two *external* interfaces (E) or, if application-specific enhancements are employed, simply *at* a single interface. This is summarized in section 8.1.2.1 below.

Finally, to provide the end-to-end service between the two E interfaces, each I-PEP peer (i.e., each gateway) needs to provide a certain set of functions when processing and forwarding data between its local interfaces I and E (internal operation). A minimum set of requirements for the internal operation of an I-PEP gateway is described in section 8.1.2.2 below.

All these additional specifications are deliberately kept to a minimum to allow for flexible implementations and vendor differentiation.

### 8.1.2.1 External Interface 'E' to Application Peers

For the external interface, three layers are potentially relevant: the IP layer, the transport layer, and the application layer. An I-PEP gateway is not required to operate at the link layer (i.e., perform layer 2 bridging) so that this is not considered further.

At the IP layer, an I-PEP gateway shall be able to forward all IP packets transparently according to their destination address unless specific processing rules are defined. (Note that, in the absence of specific processing rules, an I-PEP gateway would act as an IP router.) For I-PEP gateways as per this specification, processing rules are only defined for TCP connections (as discussed further below). An I-PEP gateway shall handle other—i.e., non-TCP—traffic as follows:

- Non-TCP traffic received by the I-PEP server SHOULD be routed transparently through the I-PEP gateway. This includes (but is not limited to) IGMP, ICMP, UDP, among others. An I-PEP gateway MAY selectively discard (filter) packets per its configuration. No performance enhancement needs to be applied to such traffic.
- I-PEP gateways MAY act as routers and need not perform re-assembly of IP packets that were fragmented on the way (e.g., for transmission across the satellite link).
- IPsec encrypted TCP connections MUST be forwarded transparently through the I-PEP system. (Note that the I-PEP gateway may not even notice that a TCP payload is contained in the respective packets).

At the transport layer, an I-PEP gateway SHOULD be able to detect TCP packets and MUST support enhancement of TCP connections as follows if only TCP enhancement is active and no application-specific support is provided (for application-specific support see below):

- TCP connections running via an I-PEP system MUST reach their ultimate destination as specified in the destination IP address by the initiating application peer.
- TCP connections MAY be terminated transparently for the application peers, MAY need to be explicitly addressed to the I-PEP gateway (acting as a proxy) by the application peers, or MAY employ other protocol mechanisms to identify application TCP connection to be acted upon. All approaches are equally valid.
- Reliable data exchange MUST be provided between application peers. Data sent by one application MUST NOT be lost, altered, reordered, or duplicated. That is, the perceived reliability semantics of TCP MUST be preserved.
- As noted above, I-PEP entities may be set up in a symmetric (peer-peer) or asymmetric (client-server) style. A symmetric I-PEP peer MUST provide enhancements for TCP

connections in both ways, asymmetric I-PEP peers MAY but need not support bi-directional connection setup.

- Data markings by the TCP URGENT pointer MAY (but need not) be preserved end-to-end across the performance enhancement. When preserved, the TCP URGENT pointer MUST point to the same byte of data as the as received from the application TCP connection.
- TCP options and flags MAY be preserved end-to-end. It is up to the I-PEP gateway to determine which options to pass on and which not. The I-PEP gateways are responsible for performing any translations as necessary. For example, if, as a result of non-end-to-end option negotiation, two application peers use different TCP parameters (e.g., MSS) it is up to the two I-PEP gateways to provide for the necessary buffering and adjustment.
- Depending on the mode of operation of the I-PEP gateway (e.g, when acting as a proxy), TLS-protected TCP connections MAY be forwarded transparently (i.e., without TCP enhancement).

At the application layer, the I-PEP peer MAY act as an *intermediary* (e.g., an HTTP proxy) and perform additional functions beyond plain TCP enhancement (e.g., HTTP prefetching). As the current version of this specification does not define application layer functionality, we provide only general statements regarding this mode of operation:

- An I-PEP peer providing application-specific support MUST conform to the respective application protocol when communicating with the application peer(s).
- Either I-PEP peer MAY terminate an application TCP connection and respond locally to the respective application's message if the I-PEP peer has sufficient knowledge to generate an appropriate response that might similarly have come from the application peer indicated as the ultimate destination. As “appropriate response” cannot be defined in technical terms, the ultimate judgment is up to the end user and her expectations.
- If one I-PEP peer needs to coordinate with the other I-PEP peer to perform application-specific enhancements, it MUST ensure that the other I-PEP also supports this enhancement function before intervening on the application TCP connection. Note that an I-PEP peer may always intercept a TCP connection and decide to fall back to plain TCP enhancement if no application-specific support is provided by its I-PEP peer.
- I-PEP gateways implementing application-specific enhancements SHOULD preserve the “end-to-end” semantics of the application TCP connection as much as possible.
- How an I-PEP gateway determines which application TCP connections to act upon is implementation dependent. I-PEP gateways MAY operate transparently to the application

peers, they MAY require explicit configuration, or they MAY make use of other protocol mechanisms.

Regardless of the protocol layer at which processing takes place, the following statements may apply for communications across two I-PEP peers:

- I-PEP gateways MAY preserve the ToS field of the IP header (used for differentiated services) or they MAY provide mapping between different diffserv classes. ToS field preservation is defined such that the ToS markings are the same for the IP packets entering the I-PEP and exiting. I-PEP gateways MAY also use a default ToS field for outgoing packets to the E interfaces.

Note that preserving the ToS field of the IP header is only possible if incoming packets are mapped one-to-one into outgoing packets (requiring identical path MTU size and the same size of IP and TCP options on both sides) since it cannot be guaranteed that the ToS field does not change across different packets belonging to a single TCP or I-PEP connection. This poses a practical constraint on such a feature.

### 8.1.2.2 Gateway Processing Requirements: Internal 'I' to External Interface 'E'

The previous section has described the general operation of an I-PEP system comprising two I-PEP peers as a whole. Apparently, similar considerations are necessary on a per-I-PEP-peer basis to achieve this overall behaviour. As many of these considerations are straightforward (and details may benefit from vendor's implementation experience), only a rough guidance is given in this subsection, again following the aforementioned protocol layering.

At the IP layer, the same considerations as above apply. It is up to (the configuration of) the respective I-PEP gateway which IP packets are captured, which are processed, which are forwarded, and which are discarded.

At the transport layer, an I-PEP gateway **MUST** be able to capture and enhance TCP connections<sup>1</sup>—transparently, as a proxy, or controlled via some other protocol. The ultimate target address of a TCP connection **MAY** be derived from the destination IP address contained in the IP packet carrying the SYN packet, **MAY** be obtained from static configuration, **MAY** be communicated as part of the application protocol (e.g., when acting as an HTTP proxy), or **MAY** be obtained via other protocol mechanisms (such as SOCKS).

As described in section 3.1.1, the end-to-end semantics of a TCP connection setup **MAY** but need not be preserved. Data transmission, reliability, and congestion control **SHOULD** be handled hop-by-hop between the I-PEP peers and between the I-PEP peers and the application peers. Flow control **SHOULD** be handled end-to-end by means of buffering and backpressure. While a TCP connection is established, data flowing in either direction **MUST** be forwarded without duplication, loss, or reordering.

The I-PEP gateway **MUST** support shared satellite links (forward channel as well as backchannel) and thereby a dynamically and potentially fast changing bandwidth for a single connection as well as for the overall bandwidth for one DVB/RCS terminal. Therefore, the implementation of Congestion Control, or some form of coordination with the available bandwidth, is mandatory.

The output of an I-PEP entity **MUST** be a SCPS-TP enhanced TCP connection consistent with the peered application's intentions. Additional TCP and IP options are made available by this document, SCPS-TP or existing or proposed Internet RFCs, and may be implemented by an I-PEP entity.

At the application layer, an I-PEP gateway is responsible for adhering to the respective application protocol specification. If application-specific enhancements are provided requiring consent from the remote I-PEP peer, the I-PEP peer needs to ensure that this support is available before acting upon an application TCP connection. For purely local enhancement, the I-PEP gateway may

---

<sup>1</sup> Support for other transport and control protocols is beyond the scope of this document.

perform any actions on the application TCP connection (terminating it, responding to requests, etc.) deemed necessary to best serve the application peer's intentions.

Either or both I-PEP peer MAY open additional I-PEP transport or TCP connections to each other and/or the application peers to perform additional actions when necessary.

In any case, the I-PEP performance with respect to (future) application-specific enhancements SHOULD only be measured against the application's performance as perceived by the user. But the I-PEP gateways SHOULD respect privacy of data conveyed to and from users.

## ***8.2 Management***

There is no generalized management framework for an I-PEP currently specified.

## 9 ADAPTABILITY AND EXTENSIBILITY

The I-PEP specification is designed to provide basic interoperability between implementations / devices from different vendors while at the same time allowing for later ESA-defined and particularly for vendor-specific extensions in many ways. The latter is crucial for a number of reasons including (but not limited to):

- Most important, vendors have specific skills developed over time that provide a competitive edge. The I-PEP specification must preserve this while ensuring a base level of interoperability.
- Any standardization process is naturally less reactive than the market requires so that a vendor must have a way to satisfy customer demands quickly without being required to invent a separate protocol on their own.
- The use of a TCP based solution allows other enhancements drawn from the Internet community, in particular research on other wireless communications, to be incorporated into implementations.
- New features being developed require experience, experiment, and extensive real-world testing – rather than from scratch round-table standardization. Vendors must have the ability to implement and continuously develop new features of their systems based upon the I-PEP protocol specification before those “proprietary specifications” may get frozen and possibly integrated into e.g. a future release of the I-PEP specification.

With this strong need for extensions in mind, the present I-PEP specification caters for extensibility at several levels:

- 1) A vendor identifier—the VendorID—will be defined as a 16 bit code for which any manufacturer of satellite software products can register with ESA (and will be assigned a unique identifier). With a reserved value, this identifier can be extended in case more than 64K identifiers would be registered.
- 2) Vendor identifiers, product codes, and version numbers are exchanged upon I-PEP Session setup so that the communicating entities get an idea who their respective peers are and what functionality they provide. This is also of importance for ensuring backward compatibility e.g. within the product line of a single vendor.
- 3) Vendor-specific extensions (PDU types, capabilities, parameters, reason codes, etc.) are possible in conjunction with I-PEP Session setup and negotiation allowing vendors to introduce entire new features in a virtually arbitrary way into the course of an I-PEP Session which can then be inherited by (future) I-PEP transport connections.

- 4) The Vendor-specific feature list will allow I-PEP peers to indicate support for messages and functionality across several vendors, i.e. even foreign vendor features may be declared. This allows to gradual migration of vendor-specific extensions to common features if widespread market acceptance is ensured.

In addition to—and independent of—any “air interface” protocol signaling, vendor-specific algorithms may be used within each I-PEP device. This is catered for by specifying only minimal requirements on the information exchange procedures and leaving many aspects (e.g. transmission rate control, heuristics for web prefetching) up to the respective implementer.

This subsection discusses how to handle extensibility in section 9.1—particularly outlining implications from mixing extended and non-extended I-PEP systems.

**A key observation here is that a careful protocol and system design will allow even non-extended systems to benefit from communicating with extended ones—and that the extensibility concept to be pursued for I-PEP by no means implies a fallback to pure minimal baseline functionality if two devices from different vendors meet.**

Finally, subsection 9.2 provides several specific examples of how the various extensibility mechanisms of the I-PEP specification can be used in I-PEP systems.

## 9.1 *Implications of the Extensibility Concept*

As discussed in the introduction to this section, the I-PEP specification provides numerous extension mechanisms for the air interface including vendor-specific options, means for extensible feature negotiation, and particularly the freedom in the choice of algorithms.

Vendors are capable of specifying additional messages and message contents allowing their systems to perform better when talking to a peer also supporting the respective extensions. Obviously, understanding the semantics is a pre-requisite for reacting properly—which is most likely to hold for equipment from the same manufacturer.

However, it should be noted that the feature negotiation mechanism introduced in the Session and, in a limited fashion, in the transport protocol allows an I-PEP peer to declare capabilities beyond those derived from the respective product code and version number. This mechanism enables a device to indicate support for additional features defined as vendor-specific extensions by third parties. This approach in turn allows well-regarded features—that may even become a de-facto standard over time—to be smoothly migrated into all products regardless of the respective maker. Thereby, this approach paves the way to incorporation into future revisions of the I-PEP specification.

The other key element of extensibility and vendor differentiation is the very strict interpretation of the “air interface” definition taken by the draft I-PEP specification. The I-PEP specification is deliberately kept very basic when it comes to distributed algorithms between client and server devices. Only the basic rules—how to react to an incoming packet—are defined to ensure interoperability, particularly for the setup, teardown, and maintenance of I-PEP sessions and I-PEP transport connections.

Accelerated data exchange for TCP is defined supporting considerable flexibility which allows algorithms and approaches a) to be tailored to specific system environments (where possibly only the server side needs to adapt) and b) to evolve gradually over time without requiring revisions to the specification. To keep this initial revision of the I-PEP specification simple, it does not include any application-specific functionality—such as HTTP prefetching—and limits its support to including a few hooks for these features.

For data exchange itself, the packet format and rules of TCP and, to a large degree, SCPS-TP are followed with some modification. In particular, only basic rules are set forth for data transmission and handling of acknowledgements. Information useful to construct and tune a variety of control loops is available between sender and receiver—but the precise usage is entirely left to the specific implementation. This allows vendor differentiation based upon a variety of bandwidth arbitration and allocation schemes on the satellite link on the hub station (server) side. Receivers (terminals) can rely on eventually receiving the data but data rate control and retransmission / reliability strategies may widely differ—and still all work with the same client implementation. In those cases, the same (trivial) client implementation may work in an acceptable manner with a trivial

fixed bandwidth server implementation, behave well with an intelligently rate-controlled scheme, and perform even better when communicating with a predictive bandwidth allocation scheme using some kind of magic.

It is very much up to the service providers to choose the products, configurations, and policies that determine the overall system performance and to tailor them to meet their needs in the respective setting they are addressing. Nevertheless, the I-PEP specification will ensure that arbitrary client components will work in this infrastructure. And it allows that a service provider may change the server as the requirements change – without a need to replace the clients as well.

The I-PEP air interface approach taken does not just cater for cross-vendor interoperability. The minimal functional requirements imposed on a compliant I-PEP implementation also allow for a wide range of products by each individual vendor. Hence, a minimal implementation can be tailored to fit the processing power and memory constraints of a small embedded device while the same protocol is suitable to provide trunking for a regional ISP based upon (clusters of co-located) powerful devices.

Finally, vendors may provide further differentiation through local system integration while maintaining interoperability with other vendors' servers and clients. For example, some vendors may decide to provide transparent capturing while this may not be suitable in other scenarios. Different vendors may follow different approaches for system configuration, maintenance, and management. Similarly, authentication, authorization, accounting, and billing interfaces may differ.

In summary, the baseline I-PEP air interface specification is expected to foster interoperability in a way that allows for crucial vendor differentiation in many respects while still providing benefits even in cross-vendor deployments. On one hand, this puts service providers into a position to choose the appropriate server system(s) for their respective service platform(s) while not being restricted in their choice for the terminal side, again with the option to switch server systems as requirements and services change over time. On the other hand, this enables vendors to specialize in certain areas, team up with other vendors in others, thereby enlarging the markets they are able to address. While putting them at increased competition in their established installation (for new clients or because of the threat server replacement), at the same time, gaining access to new platforms (so far owned by other vendors) is much simplified.

The following subsection gives a few examples on the use of extension mechanisms within the I-PEP framework.

## 9.2 *Examples for Vendor-specific Extensions*

This brief section is intended to give some guidance by means of some short examples on how vendor-specific extensions may be incorporated to realize features beyond the scope of the present I-PEP specification. It should be noted that the aspects all four examples (and further extension mechanisms) can easily be combined.

For the following examples, we assume a vendor VenEx who has been assigned the Vendor ID 323. The vendor has also defined appropriate product and version codes and thus one of their devices can determine based upon the information exchanged during the I-PEP session setup whether or not it is communicating with a peer also developed by VenEx or not and, if so, which functionality the two devices can agree upon.

### 9.2.1 EXAMPLE 1: APPLICATION-LAYER-SPECIFIC EXTENSION

Two I-PEP peers from VenEx may negotiate, e.g., as part of an I-PEP Session setup exchange, that they support (a certain variant) of HTTP prefetching. Thereby, they establish a Session context (Session\_ID) with HTTP prefetching capability to be indicated by a (standardized) Service Tag (HTTP\_Prefetch\_Tag) within the Session. The precise HTTP prefetching parameters (assuming there are some) are also agreed upon.

With this knowledge, subsequent HTTP requests for web pages passing through the two I-PEP peers are acted upon and HTTP prefetching functionality is invoked by having the newly established I-PEP transport connection make reference to the Session\_ID and the HTTP\_Prefetch\_Tag in the I-PEP extended option *Session ID and Service Tag*.

### 9.2.2 EXAMPLE 2: SESSION PROTOCOL EXTENSION

Assume that VenEx, which makes use of session-oriented handling per peer I-PEP, has developed a new software for their client and server devices that is capable of continuing servicing a customer while one server goes down, e.g. for maintenance purposes or becomes overloaded. VenEx has now defined two additional vendor-specific session control PDUs: VenEx\_SWITCH and VenEx\_SWITCH\_ACK.

The server device uses VenEx\_SWITCH to indicate to a client device that it is supposed to immediately connect to one particular other server device (to which this server device has already transferred the client device's state for seamless operation). The client device acknowledges receipt and successful switch-over (after contacting the indicate new server device) using a SWITCH\_ACK.

If both client and server device are supplied by VenEx they are able to use this feature (which they may have negotiated up front during Session setup). If the client device does not support it, the

server device will receive no response to its SWITCH and ultimately give up. If the server device does not support the extension, the client device will simply not notice because it never receives such a PDU.

### 9.2.3 EXAMPLE 3: PARAMETER EXTENSION

Assume that a vendor wants to provide additional security by authenticating all transport connections from client devices using public key cryptography algorithm and therefore run TLS on top of a TCP transport. In this case, the vendor can define an additional TCP option to indicate that TLS shall be used on top of I-PEP transport for this transport connection between the two I-PEP peers; this option is named `VenEx_USE_TLS` and is drawn from this vendor's SatLabs-assigned *Vendor ID*, as described by 5.2.7. If one I-PEP peer wants to secure an I-PEP transport connection, it includes `VenEx_USE_TLS` in the SYN packet to its peer and if the response also indicates this option, the two entities can go ahead and initiate TLS-based authentication and setup of a shared security context.

If two devices communicate that both support this extension they can make use of the extended transport connection.

It should be noted that this—artificial—example is rather inefficient since the use of TLS incurs additional round-trips to establish the TLS security context which, to some degree, defeats the purpose of performance enhancement to reduce sensibility to RTT.

### 9.2.4 EXAMPLE 4: ALGORITHM EXTENSIONS

The I-PEP specification only defines the air interface between two I-PEP peers and leaves local algorithms up to the implementation. Assuming an I-PEP sender capable of determining the available link bandwidth dynamically and most accurately (e.g., by using the manufacturer VenEx's proprietary rate prediction device `"/dev/crystal-ball"`), an I-PEP equipped with such a device will not need to rely on ACK clocking or ECN for determine the allowed transmission rate.

Therefore, the I-PEP entity may signal to its peer that it does not rely on ACK for clocking, utilizing the ACK Frequency Reduction (AFR) in the opposite direction, and that it does not support ECN (and hence the peer need not echo congestion indications). The algorithm and the proprietary crystal ball implementation need not be advertised as they do not impact the operation between the two peers.

For the sending side, the I-PEP entity will follow the suggestions from crystal ball device and adapt its transmission rate accordingly. On the receiving side, it will happily provide ACKs and/or timestamps as necessary to support its peer in its TCP operation.

### 9.2.5 SUMMARY: VENDOR EXTENSION SIGNALING

This subsection provides an example of how the vendor extensions described above would be signalled as part of the I-PEP extended TCP options. As stated above, we assume that the Vendor ID of VenEx is '323', thus requiring the extended version of the extended capabilities binding space. We further assume that the option signalling the use of TLS shall be used as indicated above and that the code point for this option is '2' in the respective vendor's specific data first octet used for independent versioning and vendor types. Moreover, we assume that the TCP connection setup is related to an HTTP connection that shall be enhanced according to the—previously negotiated as part of the Session context '0x4711'—HTTP prefetching service '0xCC'. Finally, we assume that in this particular case, the vendor also wants to include the *really cool option* indicated as option '9' in its number space—which is too large to fit into the regular TCP option space so that the TCP option data feature needs to be invoked.

Figure 20 on the following page depicts how this combination of options looks like:

In this figure, the IP header does not carry any options, the TCP header does not show any standard options and only carries those for SCPS-TP and I-PEP.

The most space efficient representation is used for the I-PEP extended option space, i.e. all extended options that fit into the regular TCP option space are accumulated in a single SCPS capability option. The I-PEP extended option set is indicated by the *second* occurrence of the SCPS option which is 15 bytes length in total.

The first I-PEP extended option is derived from the standard I-PEP binding space (100) and denotes the service tag and session Id (type 0001), indicating service tag 0xCC and context id 0x4711.

Next, follows a vendor-specific binding space: the vendor code 323 is represented as 255+68 (0xFF + 0x44) and thus makes use of the Vendor ID escape. The vendor specific option type and version indicates that TLS should be used. The vendor wants to include the *really cool option* that requires it to make use of the TCP option data. This is indicated by including the corresponding standard I-PEP option (binding space 100) of type 5 and indicating the total length of the TCP option data (30 bytes) in 16 bit words (option length=15).

The TCP header length will indicate a data offset of 44 bytes (20 bytes standard TCP header plus 24 bytes options). In this case the TCP Option of kind End of Option List and No-operation are used to pad to a 32-bit boundary as per RFC 793.

In the beginning of the TCP data portion, the first 30 bytes are reserved for options as indicated above: another SCPS option is used, again encoding the vendor-specific binding space, and then giving room for the 24 bytes of *really cool option data*. As the option data only requires 29 bytes, a reserved single byte of padding is added to achieve 16-bit alignment.

0	1	2	3	4	5	6	7	
Standard IP Header								IP header octet 1
								IP header octet 20
Standard TCP Header								TCP header octet 21
								TCP header octet 40
SCPS Capabilities (20)								TCP option octet 1
Length (=4)								TCP option octet 2
Regular SCPS Capabilities								TCP option octet 3
								TCP option octet 4
SCPS Capabilities (20)								TCP option octet 5
Length (= 18)								TCP option octet 6
Extended Option Binding Space = 100								TCP option octet 7
len=3 -> 6 octets   0   0   0   1								TCP option octet 8
Service Tag (= 0xCC)								TCP option octet 9
Session ID (= 0x4711)								TCP option octet 10
								TCP option octet 11
reserved = 0								TCP option octet 12
Extended Option Binding Space = 100								TCP option octet 13
len=3 -> 5 octets   1   0   0   0								TCP option octet 14
Vendor ID Escape (0xFF)								TCP option octet 15
Vendor ID' (0x44)								TCP option octet 16
Vendor specific, TLS version 2 = 2								TCP option octet 17
No Operation (pad) = (0x1)								TCP option octet 18
Extended Option Binding Space = 100								TCP option octet 19
len=2 -> 4 octets   0   1   0   1								TCP option octet 20
Option data length = 15 (-> 30 octets)								TCP option octet 21
reserved = 0								TCP option octet 22
End of Options List = (0x0)								TCP option octet 23
No Operation (pad) = (0x1)								TCP option octet 24
=====								(TCP data begins here)
SCPS Capabilities (20)								data option octet 1
Length (= 30)								data option octet 2
Extended Option Binding Space = 100								data option octet 3
len=14 -> 28 octets   1   0   0   0								data option octet 4

---

	Vendor ID Escape (0xFF)		data option octet 5
+-----+		+-----+	
	Vendor ID' (0x44)		data option octet 5
+-----+		+-----+	
	here comes the really cool option data		data option octet 6
~	(24 bytes)	~	data option octet 29
+-----+		+-----+	
	reserved = 0		data option octet 30
+-----+		+-----+	

---

Figure 20: Complete example of SCPS-TP and I-PEP options

## 10 REFERENCES

[RFC3135] RFC3135 Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations

[SatLabs I-PEP] Miscellaneous SatLabs contributions as on e-project directory for SatLabs

[SatLabs] [www.satlabs.org](http://www.satlabs.org)

[I-PEP] Draft Specification as contained in Part II of this document.