Digital Transmission Licensing Administrator

DTCP Volume 1 Supplement E Mapping DTCP to IP (Informational Version)

Hitachi Maxell, Ltd.
Intel Corporation
Panasonic Corporation
Sony Corporation
Toshiba Corporation

Revision 1.4 ED5
April 8, 2014

Preface

Notice

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. Hitachi Maxell, Ltd, Intel, PANASONIC, Sony, and Toshiba (collectively, the "5C") disclaim all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

Some portions of this document, identified as "Draft" are in an intermediate draft form and are subject to change without notice. Adopters and other users of this Specification are cautioned these portions are preliminary, and that products based on it may not be interoperable with the final version or subsequent versions thereof.

Intellectual Property

Implementation of this specification requires a license from the Digital Transmission Licensing Administrator.

Contact Information

Feedback on this specification should be addressed to dtla-comment@dtcp.com.

The Digital Transmission Licensing Administrator can be contacted at dtla-manager@dtcp.com.

The URL for the Digital Transmission Licensing Administrator web site is: http://www.dtcp.com.

Printing History:

2004-01-07	DTCP Volume 1 Supplement E Revision 1.0
2005-02-28	DTCP Volume 1 Supplement E Revision 1.1
2007-06-15	DTCP Volume 1 Supplement E Revision 1.2
2010-03-19	DTCP Volume 1 Supplement E Revision 1.3
2010-09-10	DTCP Volume 1 Supplement E Revision 1.31
2011-12-14	DTCP Volume 1 Supplement E Revision 1.4
2012-01-11	DTCP Volume 1 Supplement E Revision 1.4ED1
2012-04-23	DTCP Volume 1 Supplement E Revision 1.4ED2
2013-06-05	DTCP Volume 1 Supplement E Revision 1.4ED3
2014-05-21	DTCP Volume 1 Supplement E Revision 1.4ED4

TABLE OF CONTENTS

FIGURES

TABLES

V1SE 1 Introduction

This supplement describes the mapping of DTCP onto Internet Protocol (IP). All aspects of IEEE 1394 DTCP functionality except those described in Appendix D of Volume 1 which do not apply to this mapping is preserved and this supplement only details DTCP-IP specific changes or additions.

V1SE 1.1 Related Documents

This specification shall be used in conjunction with the following publications. When these publications are superseded by an approved revision, the revision shall apply.

- Digital Transmission Content Protection Specification Volume 1 and Volume 2
- FIPS 197 ADVANCED ENCRYPTION STANDARDS (AES), November 26, 2001
- NIST Special Publication 800-38A 2001 Edition, Recommendation for Block Cipher Modes of Operation, Methods and Techniques,
- RFC768 User Datagram Protocol
- RFC791 Internet Protocol
- RFC793 Transmission Control Protocol
- RFC1945 Hypertext Transfer Protocol – HTTP/1.0
- RFC2616 Hypertext Transfer Protocol – HTTP/1.1
- RFC1889 RTP: A Transport Protocol for Real-Time Applications
- UPnP ContentDirectory:2, ContentDirectory:2 Service Template Version 1.01, UPnP Forum, May 31, 2006.

V1SE 1.2 Terms and Abbreviations

DTCP-IP	DTCP volume 1 Supplement E
DTCP Socket	Socket used for AKE commands
E-EMI	Extended Encryption Mode Indicator
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
PCP	Protected Content Packet
RTP	Real-time Transport Protocol
RTT	Round Trip Time
Socket	IP-address concatenated with port number [e.g. <host>:<port>]
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
PCP2	Protected Content Packet 2
PCP-UR	Protected Content Packet – Usage Rule

V1SE 1.3 Treatment of Portions of the Specification marked "NOT ESTABLISHED"

Features of this specification that are labeled as "NOT ESTABLISHED" describe capabilities the usage of which has not yet been implemented or established by the 5C.

V1SE 2 Modifications to 4.2.3.2 Extended Format Fields (Optional Components of the Device Certificate)

For IP, the optional content channel cipher for AES-128 is not used.

V1SE 3 Modifications to Chapter 5 Restricted Authentication

Restricted authentication is not permitted for DTCP-IP transports.

V1SE 4 Modifications to Chapter 6 Content Channel Management Protection

V1SE 4.1 Modifications to 6.2.1.1 Exchange Keys

DTCP-IP requires only a single exchange key for all defined E-EMI.

V1SE 4.2 Modifications to 6.2.1.2 Session Exchange Keys (K_S)

For transaction based Move function the Session Exchange Key shall not be used.

V1SE 4.3 Modifications to 6.2.2.2 K_C for AES-128

The Content Key (K_C) is used as the key for the content encryption engine. K_C is computed from the three values shown below:

- Exchange Key K_X where only a single exchange key is used for all E-EMIs to protect the content. Note for the following formulas, that K_S is used instead of K_X when Session Exchange Key is used, K_R is used instead of K_X when Remote Exchange key is used, K_{XM} is used instead of K_X when Move Exchange Key is used, and K_{XH0} is used instead of K_X/K_{XM} when DTCP_Descriptor is used and DOT is asserted.
- Seed for content channel N_C generated by the source device which is sent in plain text to all sink devices.
- Constant value C_{A0}, C_{B1}, C_{B0}, C_{C1}, C_{C0}, or C_{D0} which corresponds to an E-EMI value in the packet header.

The Content Key is generated as follows:

K_C = J-AES(K_X, f[E-EMI], N_C) Where:

f[E-EMI] {
f[E-EMI]=C_{A0} when E-EMI = Mode A0
f[E-EMI]=C_{B1} when E-EMI = Mode B1
f[E-EMI]=C_{B0} when E-EMI = Mode B0
f[E-EMI]=C_{C1} when E-EMI = Mode C1
f[E-EMI]=C_{C0} when E-EMI = Mode C0
f[E-EMI]=C_{D0} when E-EMI = Mode D0
}

C_{A0}, C_{B1}, C_{B0}, C_{C1}, C_{C0}, and C_{D0} are universal secret constants assigned by the DTLA. The values for these constants are specified in DTCP Specification available under license from DTLA.

Additional rules for AES-128 Cipher are described in the DTCP Specification available under license from the DTLA.

V1SE 4.3.1 Modifications to 6.2.2.2.1 AES-128 Related Key and Constant Sizes

Followings are the lengths of the keys and constants described above:

Key or Constant	**Size** (bits)
Exchange Key (K_X)	96
Session Exchange Key (K_S)	96
Remote Exchange Key (K_R)	96
Scrambled Exchange Key (K_{SX})	96
Constants (C_{A0}, C_{B1}, C_{B0}, C_{C1}, C_{C0}, C_{D0})	96
Initial Vector Constant (IV_C) see V1SE 4.25	64
Content Key for AES-128 Baseline Cipher (K_C)	128
Seed for Content Channel (N_C)	64

Table 1 Length of Keys and Constants (Content Channel Management)

V1SE 4.4 Modifications to 6.2.3.2 AK_C for AES-128

When encrypting and decrypting a PCP2, the Alternate Content Key (AK_C) shall be used as the key for the content encryption engine. AK_C is computed from the four values shown below:

- Exchange key K_X where only a single exchange key is used for all E-EMIs to protect content. Note for the following formulas, that K_S is used instead of K_X when Session Exchange Key is used, K_R is used instead of K_X when Remote Exchange Key is used, and K_{XM} is used instead of K_X when Move Exchange Key is used.
- Content Management Information (CMI) which includes both the Header and Body field of CMI packet. The format of CMI packet is described in V1SE 4.26.2.1.
- A random number N_C generated by the source device using RNG_F which is sent in plain text to all sink devices in PCP2(s).
- Constant value C_{A0}, C_{B1}, C_{B0}, C_{C1}, C_{C0}, or C_{D0} which corresponds to an E-EMI value in the packet header.

The Alternate Content Key is generated as follows:

AK_C = J-AES(K_{XH}, f[E-EMI], N_C) Where:
K_{XH} = [SHA-1(K_X || CMI)] $_{lsb\_96}$
f[E-EMI] {
f[E-EMI]=C_{A0} when E-EMI = Mode A0
f[E-EMI]=C_{B1} when E-EMI = Mode B1
f[E-EMI]=C_{B0} when E-EMI = Mode B0
f[E-EMI]=C_{C1} when E-EMI = Mode C1
f[E-EMI]=C_{C0} when E-EMI = Mode C0
f[E-EMI]=C_{D0} when E-EMI = Mode D0
}

C_{A0}, C_{B1}, C_{B0}, C_{C1}, C_{C0}, and C_{D0} are universal secret constants assigned by the DTLA. The values for these constants are specified in DTCP Specification available under license from DTLA.

V1SE 4.5 Modifications to 6.3.1 Establishing Exchange Keys

Source devices assign a random value for the Session Exchange Key (K_S) and the Remote Exchange Key (K_R) (using RNG_F) being established.

The method used to scramble the Remote Exchange Key (K_R) in the RA-AKE is the same as that used to scramble the Exchange Key (K_X) specified in 6.3.1

It is mandatory that source devices expire all Exchange Keys (i.e. Exchange Key (K_X) Session Exchange Key (K_S), and Remote Exchange Key (K_R)) within 2 hours after all content transmission using PCP(s) and PCP2(s) has ceased.

It is mandatory that sink devices expire all Exchange Keys (i.e. Exchange Key (K_X), Session Exchange Key (K_S), and Remote Exchange Key (K_R)) within 2 hours of continuous non-use of any Exchange Key for decryption.

Source and sink devices must expire their Exchange Keys when they detect themselves being disconnected from all mediums. For wireless mediums this means when device detects that it is not connected to an access point or it is not directly connected to another device.

Expiration of K_R by source device based on the above rules shall be done even while a K_R keep-alive timer for the K_R is counting (see V1SE 10.7.2 and **Error! Reference source not found.**).

Source devices shall not change or expire Exchange Key (K_X) or Session Exchange Key (K_S) during content transmission using PCP(s) or PCP2(s). However source devices may change or expire Exchange Key (K_X) or Session Exchange Key (K_S) during content transmission for Remote Access only. Source devices shall not change or expire Remote Exchange Key during content transmission for Remote Access using PCP(s) or PCP2(s).

V1SE 4.6 Modifications to 6.3.3 Establishing Content Keys

This section replaces section 6.3.3 and describes the mechanism for establishing the Content Keys (K_C) used to encrypt/decrypt content being sent over DTCP-IP.

Source devices that do not support PCP-UR generate N_C as follows:

- For RTP transfers, source devices generate a 64 bit random number as an initial value for N_C using RNG_F. N_C is updated periodically by incrementing it by 1 mod 2^{64} during RTP transmission while PCP or PCP2 is in progress regardless of the value of E-EMI. The same value of N_C shall be used for all RTP simultaneous transmissions. The minimum period for update of the N_C is defined as 30 seconds, and the maximum period is defined as 120 seconds.
- For HTTP transfers, source devices generate a 64 bit random number as an initial value of N_C for the initial TCP connection using RNG_F. The initial N_C for subsequent TCP connections must be different (another random number may be generated). If a HTTP response / request has more than 128 MB of content, N_C shall be updated every 128MB. N_C is updated by incrementing it by 1 mod 2^{64}. When plural HTTP responses / requests are transmitted using the same TCP connection, N_C for subsequent HTTP response / request shall be updated from the latest N_C for the TCP connection.

Source devices that do support PCP-UR understand that N_C consists of two fields; a 16 bit PCP-UR field and a 48 bit SN_C nonce, where SN_C is handled in manner similar to the 64 bit N_C nonce except that the initial value of SN_C consists of one zero followed by a 47 bit random number and is updated by incrementing it by 1 mod 2^{48}.

V1SE 4.7 Modifications to 6.3.4 Odd/Even Bit

The Odd/Even Bit is not used in DTCP-IP as N_C value is sent with each PCP or PCP2.

V1SE 4.8 Modifications to 6.4.1 Embedded CCI

Embedded CCI is carried as part of the content stream. Many content formats including MPEG have fields allocated for carrying the CCI associated with the stream. The definition and format of CCI is specific to each content format. Information used to recognize the content format should be embedded within the content.

In the following sections, Embedded CCI is interpreted to one of four states Copy Never (CN), Copy One Generation (COG), No More Copies (NMC) or Copy Freely. Copy Freely has two variations; Copy freely with EPN asserted (CF/EPN) and Copy freely with EPN unasserted (CF).

Since the rules for recording differ based on content type, COG is identified as either Copy One Generation for audiovisual content (COG-AV) or Copy One Generation for audio content (COG-Audio) in the following sections.

V1SE 4.9 PCP-UR

PCP-UR is used as a common way to carry usage rule such as APS and ICT in the PCP header. The format of PCP-UR is described in section V1SE 4.26.1.1.

PCP-UR may be used in two cases. If PCP-UR is used for content which has Embedded CCI, sink functions which do not recognize the Embedded CCI (Format-non-cognizant sink and recording function) can use information in the PCP-UR along with E-EMI.

If PCP-UR is used for content which has no Embedded CCI, sink devices can regard the PCP-UR along with E-EMI as the Embedded CCI. For this type of content, sink functions and recoding functions which recognize E-EMI and PCP-UR behave as Format-cognizant functions.

V1SE 4.10 Modifications to 6.4.1.2 Content Management Information (CMI)

Content Management Information (CMI) is a media format agnostic method to carry enhanced usage rules. CMI is transmitted by a CMI packet in the same connection as the DTCP protected packet.

The format of a CMI packet is described in V1SE 4.26.2.1. When source devices send usage rules using a CMI packet, then the DTCP protected content shall be sent via PCP2. The format of PCP2 is described in V1SE 4.26.2.2.

V1SE 4.11 Modifications to 6.4.2 Encryption Mode Indicator (EMI)

E-EMI Mode	E-EMI Value	Description
Mode A0	1100_2	Copy-never (CN)
Mode B1	1010_2	Copy-one-generation (COG) [Format-cognizant recording only]
Mode B0	1000_2	Copy-one-generation (COG) [Format-non-cognizant recording permitted]
Mode C1	0110_2	Move [Audiovisual], Copy-count
Mode C0	0100_2	No-more-copies (NMC)
Mode D0	0010_2	Copy-free with EPN asserted (CF/EPN)
N.A.	0000_2	Copy-free (CF)
	$----_2$	All other values reserved

Table 2 E-EMI Mode and E-EMI Descriptions

V1SE 4.12 Modifications to 6.4.3 Relationship between Embedded CCI and EMI

E-EMI	Embedded CCI					
	CF	CF/EPN	NMC	COG-AV	COG-Audio	CN
Mode A0 (CN)	Allowed	Allowed	Allowed[1]	Allowed	Allowed	Allowed
Mode B1 (Format cognizant only recordable)	Allowed	Allowed	Prohibited	Allowed	Allowed	Prohibited
Mode B0 (Format non-cognizant recordable)	Allowed	Allowed	Prohibited	Allowed	Prohibited	Prohibited
Mode C0 (NMC)	Allowed	Allowed	Allowed	Allowed	Allowed	Prohibited
Mode D0 (CF/EPN)	Allowed	Allowed	Prohibited	Prohibited	Prohibited	Prohibited
N.A.	Allowed	Prohibited	Prohibited	Prohibited	Prohibited	Prohibited

Table 3 Relationship between E-EMI and Embedded CCI

V1SE 4.13 Modifications to 6.4.4.1 Format-cognizant source function

Embedded CCI of programs					E-EMI
CF	CF/EPN	NMC	COG-AV	CN	
Don't care	Don't care	*[2]	Don't care	Present	**Mode A0**
Don't care	Don't care	Cannot be present	Present	Cannot be present	**Mode B1**
Don't care	Don't care	Cannot be present	Present	Cannot be present	**Mode B0**
Don't care	Don't care	Present	Cannot be present[3]	Cannot be present	**Mode C0**
Don't care	Present	Cannot be present	Cannot be present	Cannot be present	**Mode D0**
Present	Cannot be present	Cannot be present	Cannot be present	Cannot be present	**N.A.**
Other combinations					**Transmission Prohibited**

Table 4 Format-Cognizant Source Function CCI handling

V1SE 4.14 Modifications to 6.4.4.2 Format-non-cognizant source function

E-EMI or recorded CCI[4] of source content	E-EMI used for transmission
Copy Never	**Mode A0**
COG: Format cognizant only recordable	**Mode B1**
COG: Format non-cognizant recordable	**Mode B0**
No-more-copies	**Mode C0**
EPN asserted Copy Free	**Mode D0**
Copy-Free	**N.A.**

Table 5 Format-Non-Cognizant Source Function CCI handling

[1] Not typically used.

[2] Don't care, but not typically used.

[3] This combination is allowed for format-non-cognizant source function, but is not permitted for format-cognizant source function.

[4] Recorded CCI is copy control information that is not embedded in the content program and does not require knowledge of the content format to extract.

V1SE 4.15 Modifications to 6.4.4.3 Format-cognizant recording function

E-EMI	Embedded CCI for each program				
	CF	CF/EPN	NMC	COG-AV	CN
Mode A0	Recordable	Recordable	Do not record	*[5]	Do not record
Mode B1	Recordable	Recordable	Discard entire content stream[6]	*[5]	Discard entire content stream[6]
Mode B0	Recordable	Recordable	Discard entire content stream[6]	*[5]	Discard entire content stream[6]
Mode C0	Recordable	Recordable	Do not record	Do not record	Discard entire content stream[6]
Mode D0	Recordable	Recordable	Discard entire content stream[6]	Discard entire content stream[6]	Discard entire content stream[6]

Table 6 Format-cognizant recording function CCI handling

V1SE 4.16 Modifications to 6.4.4.4 Format-cognizant sink function

E-EMI	Embedded CCI for each program				
	CF	CF/EPN	NMC	COG-AV	CN
Mode A0	Available for processing	Available for processing	Available for processing[1]	Available for processing	Available for processing
Mode B1	Available for processing	Available for processing	Discard entire content stream[7]	Available for processing	Discard entire content stream[7]
Mode B0	Available for processing	Available for processing	Discard entire content stream[7]	Available for processing	Discard entire content stream[7]
Mode C0	Available for processing	Available for processing	Available for processing	Available for processing[8]	Discard entire content stream[7]
Mode D0	Available for processing	Available for processing	Discard entire content stream[7]	Discard entire content stream[7]	Discard entire content stream[7]

Table 7 Format-cognizant sink function CCI handling

V1SE 4.17 Modifications to 6.4.4.5 Format-non-cognizant recording function

E-EMI of the received stream	Recorded CCI[9] to be written onto user recordable media
Mode A0	Stream cannot be recorded
Mode B1	Stream cannot be recorded
Mode B0	No-more-copies
Mode C0	Stream cannot be recorded
Mode D0	EPN asserted Copy Free

Table 8 Format-non-cognizant recording function CCI handling

V1SE 4.18 Modifications to 6.4.4.6 Format-non-cognizant sink function

For this function, which does not recognize Embedded CCI, the content must be treated in a manner consistent with its E-EMI. For Example, treatment that does not depend on Embedded CCI is possible.

[5] If the recording function supports recording a CCI value of No-more-copies then the CCI value of No-more-copies shall be recorded with the program. Otherwise the CCI of Copy-never shall be recorded with the program.

[6] If the function detects this CCI combination among the programs it is recording, the entire content stream is discarded.

[7] If the function detects this CCI combination among the programs, the entire content stream is discarded.

[8] If the device has a rule for handling No-more-copies, this program shall be handled according to the rule. Otherwise the program shall be handled as Copy Never.

[9] Recorded CCI is copy control information that is not embedded in the content program and does not require knowledge of the content format to extract.

V1SE 4.19 Modifications to 6.4.5.1 Embedded CCI for audio transmission

Value and Abbreviation	Meaning
11	Not defined
10 (COG-audio)	Copy-permitted-per-type
01 (NMC)	No-more-copies
00 (CF)	Copy-free

Table 9 Audio Embedded CCI Values

V1SE 4.20 Modifications to 6.4.5.3 Audio-format-cognizant source function

Embedded CCI of programs			E-EMI
CF	NMC	COG-audio	
Type specific[10]			**Mode A0**
Don't care	Cannot be present	Present	**Mode B1**
Don't care	Present	Don't care	**Mode C0**
Present	Cannot be present	Cannot be present	**N.A.**

Table 10 Audio-format cognizant source function CCI handling

V1SE 4.21 Modifications to 6.4.5.5 Audio-format-cognizant recording function

E-EMI	Embedded CCI of Program		
	CF	NMC	COG-audio
Mode A0	Recordable	Do not record	Recordable[11]
Mode B1	Recordable	Discard entire content stream[12]	Recordable[11]
Mode C0	Recordable	Do not record	Recordable[11]

Table 11 Audio-format-cognizant recording function CCI handling

V1SE 4.22 Modifications to 6.4.5.6 Audio-format cognizant sink function

E-EMI	Embedded CCI of program		
	CF	NMC	COG-audio
Mode A0	Available for processing	Available for processing	Available for processing
Mode B1	Available for processing	Discard entire content stream[12]	Available for processing
Mode C0	Available for processing	Available for processing	Available for processing

Table 12 Audio-format-cognizant sink function CCI handling

V1SE 4.23 Modifications to 6.4.5.8 Audio-Format-non-cognizant sink function

Audio-format-non-cognizant sink functions shall behave as described in section V1SE.4.18.

V1SE 4.24 Modifications to 6.6.1 Baseline Cipher

For IP, the baseline cipher is AES-128 using the Cipher Block Chaining (CBC). AES-128 is described in FIPS 197 dated November 26, 2001 and the CBC mode is described in NIST SP 800-38A 2001 Edition.

V1SE 4.25 Modifications to 6.6.2.1 AES-128 Cipher

For AES-128, Cipher Block Chaining (CBC) is used. AES-128 is described in FIPS 197 dated November 26, 2001 and the CBC mode is described in NIST SP800-38A 2001 Edition. Additional rules for AES-128 Cipher are described in the DTCP specification available under license from DTLA.

[10] Usage is specified for each Audio type in Appendix A.

[11] The CCI value of No-more-copies shall be recorded with the program. Additional rules for recording are specified by each audio application in Appendix A.

[12] If the function detects this CCI combination among the programs it is recording the entire content stream is discarded.

V1SE 4.26 Modification to 6.6.3 Content Encryption Formats

V1SE 4.26.1 Protected Content Packet (PCP)

DTCP encrypted content is sent via Protected Content Packets (PCP) where the format of the PCP is described in the following figure.

	msb							lsb
Header[0]	Packet_Type		C_A2	C_A	E-EMI			
Header[1]	exchange_key_label							
Header[2]	N_C (64 bits)							
Header[3]								
Header[4]								
Header[5]								
Header[6]								
Header[7]								
Header[8]								
Header[9]								
Header[10]	Byte length of content denoted as CL (32 bits)							
Header[11]								
Header[12]								
Header[13]								
EC[0]	Content affixed with 0 to 15 bytes of padding							
EC[1]								
EC[2]								
-								
-								
-								
EC[N-1]								

Figure 1 Protected Content Packet Format

Header [0]:

Packet_Type where for PCP, the value is set to 00_2.

C_A2 and C_A identifies the cipher_algorithm as defined in the following table.

C_A2	C_A	Meaning
0	0	Baseline Cipher (AES-128) with K_C using K_X, K_S, K_R, or K_{XM}
0	1	Baseline Cipher (AES-128) with K_C using K_{XH0}
1	0	Reserved
1	1	Reserved

Table 13 C_A2 and C_A Values

E-EMI is as defined in section V1SE 4.10.

Header [1]: Contains exchange_key_label which is described in the DTCP Specification available under license from DTLA.

Header [2..9]: Contains N_C as described in section V1SE 4.6.

Header [10..13]: Denotes byte length of content and does not include any padding bytes, where CL is less than or equal to 128 MB.

EC [0..N-1]: Represents encrypted frame[13] and there is no EC when CL is zero otherwise it is a multiple of 16 Bytes in length where N = (Int((CL-1)/16)+1)*16 where padding length is equal to N-CL and Int(X) means maximum integer less than or equal to X. The value of each padding Byte is 00_{16}.

For RTP transfers, each RTP payload is encapsulated by a single PCP when CMI is not used.

[13] Cipher Block chaining resets every PCP. The IV described in V1SE.4.25 is used in an initial step in the encryption/decryption of every PCP.

For HTTP transfers, responses / requests may contain 1 or more PCPs.

V1SE 4.26.1.1 *N_C field*

Source devices that do not support PCP-UR treat N_C as a 64 bit nonce and source devices that do support PCP-UR understand that N_C consists of two fields; a 16 bit PCP-UR field and a 48 bit SN_C nonce as shown in Figure 2.

	msb							lsb
$N_C[0]$	PCP-UR (16 bits)							
$N_C[1]$								
$N_C[2]$	SN_C (48 bits)							
$N_C[3]$								
$N_C[4]$								
$N_C[5]$								
$N_C[6]$								
$N_C[7]$								

Figure 2 Nc with PCP-UR and SN_C

Source device may support PCP-UR but if a source device supports PCP-UR it shall always transmit content with the N_C with the PCP-UR field and 48 bit SN_C nonce.

V1SE 4.26.1.2 *PCP-UR field*

The following figure shows the format of PCP-UR field:

	Msb							lsb
PCP-UR[0]	UR Mode		Content Type		APS		ICT	1_2
PCP-UR[1]	AST_{INV}	DOT_{INV}	Reserved					

Figure 3 PCP-UR Format

UR Mode field indicates how the PCP-UR is interpreted. Source devices should not change the value of UR Mode in the middle of a content transmission.

UR Mode	Meaning
00_2	No information
01_2	Content stream has Embedded CCI. PCP-UR has the same information as the Embedded CCI
10_2	Content stream has no valid Embedded CCI. PCP-UR and E-EMI are regarded as the Embedded CCI
11_2	Reserved

Table 14 UR Mode values

Content Type field indicates the type of content. Source devices should not change the value of Content Type in the middle of a content transmission. When Content Type field has value of 01_2, following APS, ICT, AST_{INV} and DOT_{INV} fields are unavailable.

Content Type	Meaning
00_2	Audiovisual
01_2	Type 1 Audio
10_2	Reserved
11_2	Reserved

Table 15 Content Type values

APS field contains analog copy protection information as described in section B.2.1 of Volume 1 of the Specification.

ICT field contains Image_Constraint_Token as described in section B.2.1 of Volume 1 of the Specification. When a source device sends multiplexed content, most restrictive value shall be set to this field.

PCP-UR[0] Bit 0 (lsb), source devices shall set to 1_2 and sink devices shall accept either 0_2 or 1_2.

AST_{INV} field contains inverted Analog_Sunset_Token as described in section B.2.1 of Volume 1 of the Specification. Where the AST_{INV} has the following meaning:

AST_{INV}	Meaning
0_2	AST-unasserted
1_2	AST-asserted

Table 16 AST_{INV}

DOT_{INV} field contains inverted value of Digital_Only_Token as described in section B.2.1 of Volume 1 of the Specification. Where the DOT_{INV} has the following meaning:

DOT_{INV}	Meaning
0_2	DOT-unasserted
1_2	DOT-asserted

Table 17 DOT_{INV}

Reserved field is the area for future extension. Source devices shall set to zero. Sink devices shall use value of Reserved field to calculate K_C in order that they can accommodate any future changes.

V1SE 4.26.1.3 *PCP-UR Capable Source Devices*

PCP-UR capable source devices shall always transmit content using the N_C that consists of the PCP-UR field and 48 bit SN_C nonce.

Source devices must provide PCP-UR when the embedded CCI does not carry any one of the following fields; APS, ICT, or Analog_Sunset_Token information.

Source devices that support PCP-UR shall support CAPABILITY_EXCHANGE subfunction and shall set PCP-UR as follows.

- Source devices shall set the UR Mode field and subsequent PCP-UR fields to zero when it transmits the following content:
  - MPEG-TS content.
  - Type 2 Audio content.
  - Multiple substreams which may have different states for Content Type and APS fields.
  - When content is received using DTCP without PCP-UR, and when the source device cannot determine the value of APS, ICT, AST_{INV} and DOT_{INV}.
- Source devices may use UR Mode 00_2 or UR Mode 01_2 when it transmits a content stream with Embedded CCI that contains CCI, APS, ICT, Analog_Sunset_Token and Digital_Only_Token information associated to that content but UR Mode 01_2 is recommended.
  - When UR Mode 00_2 is used, the source device shall set all of the fields in PCP-UR to zero.
  - When UR Mode 01_2 is used, the source device shall set the value of Content Type field according to the types of content and:
    - When value of Content Type field is 00_2 it will set the value of APS, ICT, AST_{INV} and DOT_{INV} fields equivalent to Embedded CCI.
    - When value of Content Type field is 01_2, the source device shall set APS, ICT, AST_{INV} and DOT_{INV} fields to zero.
- Source devices shall set 10_2 to the UR Mode field when it transmits content stream without Embedded CCI which determines the value of CCI, APS, ICT, Analog_Sunset_Token and Digital_Only_Token or with invalid value of such Embedded CCI. In this case, the source device shall set the value of Content Type field according to the types of content. The source device shall also set APS, ICT, AST_{INV} and DOT_{INV} fields equivalent to the information associated to the content.

- When UR Mode is 10_2, source device shall set E-EMI based on CCI of transmitting content as follows:
  Content Type 00_2 case:

E-EMI Mode	CCI
Mode A0	Copy-never (CN)
Mode B1	Copy-one-generation (COG) [Format-cognizant recording only]
Mode B0	Copy-one-generation (COG) [Format-non-cognizant recording permitted]
Mode C0	No-more-copies (NMC)
Mode D0	Copy-free with EPN asserted (CF/EPN)
N.A.	Copy-free (CF)

Table 18 E-EMI Mode and CCI mapping for Audiovisual content

In case of Move, Mode C1 of E-EMI is used.

Content Type 01_2 case:

Any content format using CCI[14] equivalent to SCMS can be transmitted as Type 1 Audio with UR Mode 10_2.

E-EMI Mode	CCI
Mode A0	N.A.
Mode B1	Copy-one-generation (COG) [Format-cognizant recording only]
Mode B0	N.A.
Mode C0	No-more-copies (NMC)
Mode D0	N.A.
N.A.	Copy-free (CF)

Table 19 E-EMI Mode and CCI mapping for Type 1 Audio content

- Source device shall set zero to the APS, ICT, AST_{INV} and DOT_{INV} fields when Content Type is 01_2.

When the DOT_{INV} field is set to one (DOT-asserted) in the PCP-UR, source functions shall use the K_{XH0} (see B.3.1) instead of K_X/K_{XM} to calculate the Content Key (K_C). Source function shall set zero and one to the C_A2 and C_A fields in the PCP, respectively while they use K_{XH0} for the Content Key.

~~K_{XH0} shall not be used for content transmission using the CMI or the Session Exchange Key or for any other purpose than the calculation of the Content Key (K_C)~~PCP-UR shall not be used with CMI.

[14] Content format without ASE-CCI can be transmitted.

V1SE 4.26.1.4 *PCP-UR Capable Sink Devices*

PCP-UR capable sink devices must confirm that the source device is PCP-UR capable by using the CAPABILITY_EXCHANGE subfunction. Sink devices can use PCP-UR only when content accompanied by the PCP-UR is encrypted by the source device which supports PCP-UR.

PCP-UR capable sink devices shall treat PCP-UR based on the value of UR Mode as follows.

UR Mode 00_2:

- Sink device shall ignore fields in PCP-UR subsequent to the UR Mode field.

UR Mode 01_2:

- If Embedded CCI is recognized, the Embedded CCI shall be used instead of PCP-UR. (Considered to be Format-cognizant sink functions and Format-cognizant recording functions.)
- If Embedded CCI is not recognized, the sink device behave as Format-non-cognizant sink functions or Format-non-cognizant recording functions and may use PCP-UR along with E-EMI to control its behavior. If content consists of multiple substreams, all the substreams are regarded as they have the same CCI with regard to the information in PCP-UR and E-EMI.
- If sink device detects value of 10_2 or 11_2 for Content Type field, it shall ignore the subsequent fields in the PCP-UR field.

UR Mode 10_2:

- Sink devices may regard the PCP-UR and E-EMI as the Embedded CCI of the content and shall disregard any embedded CCI or alternative Embedded CCI. In this case, the Sink devices behave as Format-cognizant sink functions or Format-cognizant recording functions. If a content consists of multiple substreams, all the substreams will have the same CCI.
- Sink devices may determine CCI of content from E-EMI based on the mapping shown in V1SE 4.26.1.3.
- If sink devices detect a value of 10_2 or 11_2 for Content Type field, it shall ignore the subsequent fields in the PCP-UR field and behave as a Format-non-cognizant function.

UR Mode 11_2:

- Sink device shall behave in the same way as when UR Mode is 00_2.

V1SE 4.26.2 CMI and PCP2

V1SE 4.26.2.1 *CMI Packet Format*

The format of the CMI packet is described in the following figure.

	msb							lsb
Header [0]	Packet_Type		rsv	Fixed value (11111_2)				
Header [1]	CMI_Counter							
Header [2]	Reserved (zero)							
Header [3]	Reserved (zero)							
Header [4]	Byte length of the whole CMI Field (32 bits)							
Header [5]								
Header [6]								
Header [7]								
Body [0]	CMI Field							
Body [1]								
Body [2]								
-								
-								
-								
Body [X]								

Table 20 CMI Packet Format

Header [0]: Contains **Packet_Type** field which has the fixed value of 01_2 for the CMI packet and **rsv** field which has a value of zero.

Header [1]: Contains **CMI_Counter** field.

CMI_Counter is used to inform the sink device that data in the CMI packet has changed. If a source device sends the same CMI packet or detects that the values in both the Header and Body fields of the CMI packet has not changed since the most recent packet the source device sent, the source device shall not change the value of CMI_Counter, otherwise the source device shall increase the value of CMI_Counter of new CMI packet by one. When the value of CMI_Counter reaches 255, it should wrap to zero for the next value. The initial value of the CMI_Counter should be set to a random value.

Header [2]: Contains **Reserved** field.

Header [3]: Contains **Reserved** field.

Header [4..7]: Denotes byte length of CMI Field.

Body [0..X]: Represents **CMI Field** which includes usage rules. CMI Field is described in Appendix E of Volume 1 of the Specification.

V1SE 4.26.2.2 *Protected Content Packet 2 Format*

DTCP encrypted content is sent via Protected Content Packet 2 (PCP2) when CMI packet is used to communicate usage rules. The format of the PCP2 is described in the following table.

	msb							lsb
Header [0]	Packet_Type		C_A2	C_A	E-EMI			
Header [1]	exchange_key_label							
Header [2]	N_C (64 bits)							
Header [3]								
Header [4]								
Header [5]								
Header [6]								
Header [7]								
Header [8]								
Header [9]								
Header [10]	Byte length of content denoted as CL (32 bits)							
Header [11]								
Header [12]								
Header [13]								
EC [0]	Content affixed with 0 to 15 bytes of padding							
EC [1]								
EC [2]								
-								
-								
-								
EC [N-1]								

Table 21 Protected Content Packet 2 Format

Header [0]:

Packet_Type where for PCP2 packet, the value is set to 10_2.

C_A2 and C_A identifies the cipher_algorithm as defined in the following table.

C_A2	C_A	Meaning
0	0	Baseline Cipher (AES-128) with AK_C using K_X, K_S, K_R, or K_{XM}
0	1	Prohibited
1	0	Reserved
1	1	Reserved

E-EMI is as defined in section V1SE 4.11.

Header [1]: Contains **exchange_key_label** which is described in section 8.3.4.3 of Volume 1 of the Specification.

Header [2..9]: Contains $\mathbf{N_C}$ as described in section V1SE 4.6.

Header [10..13]: Denotes byte length of content and does not include any padding bytes, where CL is less than or equal to 128MB.

EC [0..N-1]: Represents encrypted frame[15] and there is no EC when CL is zero otherwise it is a multiple of 16 Bytes in length where N = (Int((CL-1)/16)+1)*16 where padding length is equal to N-CL and Int(X) means maximum integer less than or equal to X. The value of each padding Byte is 00_{16}.

[15] Cipher Block chaining resets every PCP2. The IV described in V1SE 4.25 is used in an initial step in the encryption/decryption of every PCP2.

V1SE 4.27 Modifications to 6.7.1 Move Function

This supplement defines a Move function in addition to the one described in section 6.7.1 where content with Embedded CCI of No-more-copies content may not be remarked as Copy-one-generation but instead be transmitted as No-more-copies using Mode C1 of E-EMI for IP transport of DTCP protected content and Recording functions may record the received content without remarking embedded CCI. E-EMI Mode B1 shall be used for Move-mode when source function uses Move function described in section 6.7.1. For clarity, the move function shall be used between a single source and a single sink function.

Section V1SE 10.4 defines a protocol for transaction based Move function using Mode C1 of E-EMI, which uses Exchange key dedicated for Move.

V1SE 5 Modifications to Chapter 8 (AV/C Digital Interface Command Set Extensions)

V1SE 5.1 Modifications to 8.1 Introduction

DTCP-IP uses TCP port to send/receive DTCP control packets, status command packets, and response packets. DTCP Socket identification of source device is described in section V1SE 12.2.

Devices shall wait at least one second for a response to a command before timing out.

V1SE 5.2 Modifications to 8.3.1 AKE Control Command

This section maps the AKE control command specified in Section 8.3.1 to the DTCP-IP Control Packet Format. Except as otherwise noted, the AKE control command sub fields used with IP have the same values and functions as detailed in Chapter 8. The following format is used in both command and response frame.

	msb							lsb
Type[0]	0	0	0	0	0	0	0	1
Length[0]	(msb) Byte Length of Control Fields and AKE_Info Fields (8+N)							
Length[1]								(lsb)
Control[0]	Reserved (zero)				ctype/response			
Control[1]	Category = 0000_2 (AKE)				AKE_ID = 0000_2			
Control[2]	Subfunction							
Control[3]	AKE_procedure							
Control[4]	exchange_key							
Control[5]	subfunction_dependent							
Control[6]	AKE_label							
Control[7]	number(option)				Status			
AKE_Info[0..N-1]	AKE_Info							

Figure 4 DTCP-IP Control Packet Format

- Type, Length, and Control byte 0 are used to map DTCP to IP. Where the Type field identifies version 1 AKE control packet.
- ctype/response has the same values as referenced in chapter 8 of DTCP specification and specified by the AV/C Digital Interface Command.
- Control bytes 1..7 are identical to operand bytes 0..6 as specified in section 8.3.1, except for four most significant bits of Control byte 7 which is not used in IP.
- The data_length field form Volume 1 section 8.3.1 ~~now reflected in~~ is replaced with the Length Field, Length[0] and Length[1] to indicate the byte length of data that follows the Length field. Its value shall be 8+N for either command or response. If there are N AKE_info data bytes to be sent with the command, the command Byte Length shall be 8+N. If there are N AKE_info data bytes to be returned with the response, the response Byte Length shall be 8+N.
- The AKE_Info field is identical to the Data field specified in section 8.3.1.
- The AKE_label and source Socket of each control command should be checked to ensure that it is from the appropriate controller.

V1SE 5.3 Modification to 8.3.2 AKE status command

This section maps the AKE status command specified in Section 8.3.2 to the DTCP-IP Status Packet Format. Except as otherwise noted, the AKE status command sub fields used with IP have the same values and functions as detailed in Chapter 8. The following format is used in both command and response frame.

	Msb							lsb
Type[0]	0	0	0	0	0	0	0	1
Length[0]	(msb)	Byte length of Control Field						
Length[1]								(lsb)
Control[0]	Reserved (Zero)				ctype/response			
Control[1]	Category = 0000_2 (AKE)				AKE_ID = 0000_2			
Control[2]	Subfunction							
Control[3]	AKE_procedure							
Control[4]	exchange_key							
Control[5]	subfunction_dependent							
Control[6]	AKE_label = FF_{16}							
Control[7]	number = F_{16}				status			

Figure 5 Status Packet Format

- Type, Length, and Control byte 0 are used to map DTCP to IP. Where the Type field identifies version 1 AKE control packet.
- Ctype has the same values as referenced in Chapter 8 of DTCP specification and specified by the AV/C Digital Interface Command Set.
- Control bytes 1..7 are identical to operand bytes 0..6 as specified in Section 8.3.2.

V1SE 5.3.1 Modifications to AKE status command status field

Value	Status	Response code
0000_2	No error	STABLE
0001_2	Support for no more authentication procedures is currently available	STABLE
0111_2	Any other error	STABLE
1111_2	No information[16]	REJECTED

Table 22 AKE Status Command Status Field

[16] It is recommended that implementers do not use the "No information" response.

V1SE 5.4 Modifications to 8.3.3

V1SE 5.4.1 AKE_ID dependent field

DTCP-IP implementations only require a single exchange key, for transporting all DTCP Protected content over IP for all defined E-EMI.

For DTCP-IP, both Source and Sink shall support only Full Authentication.

Therefore Restricted Authentication procedure (rest_auth) and Enhanced Restricted Authentication procedure (en_rest_auth) are prohibited. Extended Full Authentication procedure (ex_full_auth) is NOT ESTABLISHED[17] and not used to handle Bit 3 of exchange_key field.

Bit	AKE_procedure
0 (lsb)	Prohibited
1	Prohibited
2	Full Authentication procedure (full_auth)
3	Extended Full Authentication procedure[18] (ex_full_auth, NOT ESTABLISHED[17])
4 – 7 (msb)	Reserved for future extension and shall be zero

Table 23 AKE_procedure values

V1SE 5.4.2 Modifications to Authentication selection

	Sink supported authentication procedures	
Source supported authentication Procedures	**Full_auth**	**Full_auth and Ex_full_auth**
Full_auth	Full Authentication	Full Authentication
Full_auth and Ex_full_auth	Full Authentication	Extended Full Authentication

Table 24 Authentication selection

V1SE 5.4.3 Modification to exchange_key values

For the control command, the sink device sets only one bit of this field at the start of an authentication procedure to specify which Exchange Key will be supplied by the source device after the successful completion of the procedure.

Bit	exchange_key
0 (lsb)	Prohibited
1	Prohibited
2	Prohibited
3	Exchange Key (K_X) for AES-128
4	Reserved for future extension and shall be zero
5	Session Exchange Key (K_S) for AES-128
6	Remote Exchange Key (K_R) for AES-128
7 (msb)	Reserved for future extension and shall be zero

Table 25 exchange_key values

[17] See section V1SE 1.3.

[18] Devices that support extended device certificates use the Extended Full Authentication procedure described in this chapter.

V1SE 5.5 Modifications to AKE Subfunction Descriptions

Subfunction modified for DTCP-IP are described in the DTCP specification available under license from the DTLA.

V1SE 5.6 Modifications to 8.4 Bus Reset Behavior

If the TCP connection is broken during authentication procedure, both source and sink devices shall immediately stop authentication procedure.

V1SE 6 Modifications to Appendix A (Additional Rules for Audio Applications)

V1SE 6.1 Modification to A.1 AM824 audio

Rules described in sections A.1.1, A.1.2, and A.1.2.3 are not limited to AM824 and Mode A is regarded as Mode A0 for DTCP-IP.

V1SE 6.1.1 Modification to A.1.1 Type 1: IEC 60958 Conformant Audio

Any content format with ASE-CCI equivalent to SCMS shall be regarded as Type 1 Audio.

V1SE 6.1.2 Modification to A.1.2 Type 2: DVD-Audio

Any content format containing DVD-Audio content and having ASE-CCI as described in Section A.1.2.2 shall be regarded as Type 2 Audio.

V1SE 6.1.3 Modification to A.1.3 Type 3: Super Audio CD

Any content format containing Super Audio CD content and having ASE-CCI equivalent to that described in Section A.1.3.2 shall be regarded as Type 3 Audio.

V1SE 6.2 Modification to A.2 MPEG Audio

Audio transmission via MPEG transport stream is permitted. Note that MPEG audio with ASE-CCI equivalent to SCMS is also Type 1 audio.

V1SE 7 Modification to Appendix B (DTCP_Descriptor for MPEG Transport Stream)

V1SE 7.1 Modification to B.1 DTCP_descriptor

As no standardized method for carrying Embedded CCI in the MPEG-TS is currently available, the DTLA has established the DTCP_descriptor and DTCP_audio_descriptor to provide a uniform Data field to carry Embedded CCI in the MPEG-TS. When MPEG-TS format audiovisual content is protected by DTCP, the DTCP_descriptor shall be used to deliver Embedded CCI information to sink devices.
DTCP_audio_descriptor is defined for audio transmission which uses Type 1 Audio specified in Section V1SE 6.1.1.

V1SE 7.2 Modification to B.2 DTCP_descriptor syntax

DTCP_audio_descriptor is defined for audio transmission in addition to DTCP_descriptor defined in Section B.2. The first bit value of Private_data_byte is used to distinguish DTCP_descriptor and DTCP_audio_descriptor.

In case of audio transmission, the following syntax is used, and DTCP_descriptor is referred to as DTCP_audio_descriptor.

The DTCP_audio_descriptor has the same syntax as DTCP_descriptor except for private_data_byte field. The definition of the private_data_byte field of the DTCP_audio_descriptor is as follows:

Syntax	Size(bits)	Formats
Private_data_byte{		
Descriptor_ID	1	bslbf
Reserved	5	bslbf
DTCP_CCI_audio	2	bslbf
Audio_Type	3	bslbf
Reserved	5	bslbf
}		

Table 26 Syntax of private_data_byte for DTCP_audio_descriptor

V1SE 7.2.1 Modification to B.2.1 private_data_byte Definitions:

Definition for the following fields is added for DTCP_audio_descriptor.

Descriptor_ID
This field indicates the kinds of descriptor.

Descriptor_ID	Meaning
0_2	DTCP_audio_descriptor
1_2	DTCP_descriptor

Table 27 Descriptor_ID

DTCP_CCI_audio
This field indicates the embedded CCI states for the transmission of Type 1 audio content.

DTCP_CCI_audio	Meaning
00_2	Copy-free
01_2	No-more-copies
10_2	Copy-permitted-per-type
11_2	Not defined

Table 28 DTCP_CCI_audio

Audio_type
This field indicates the Audio type.

Audio_type	Meaning
000_2	Type 1
$001_2..111_2$	Reserved for future extension

Table 29 Audio_type

V1SE 7.3 Modification to B.3 Rules for the Usage of the DTCP_descriptor

V1SE 7.3.1 Modification to B.3.1 Transmission of a partial MPEG-TS

For audio transmissions the following rules are applied:

- When a partial MPEG-TS that includes one or more programs is transmitted using DTCP, Audio-Format-cognizant source function shall insert the DTCP_audio_descriptor into the PMT[19] of each program for which ASE-CCI of Type 1 Audio is used and the ASE-CCI is not Copy-free. When the DTCP_audio_descriptor is inserted, it shall only be applied to the PMT.
- An Audio-Format-cognizant source function shall set the DTCP_CCI_audio bits according to the ASE-CCI of Type 1 Audio provided for each program within the MPEG-TS. The DTCP_audio_descriptor shall be inserted into the program_info loop of the relevant PMT.
- Additionally, if any of the Elementary Streams within a program are assigned specific ASE-CCI values of Type 1 Audio, Audio-format-cognizant source function shall set the DTCP_CCI_audio bits according to ASE-CCI of Type 1 Audio. The DTCP_audio_descriptor shall be inserted into the ES_info loop of the relevant PMT for the Elementary Stream.
- When Audio related content that is required to be treated as audiovisual content is transmitted as a part of Audio program, Audio-Format-cognizant source function, according to the upstream license, may insert DTCP_descriptor of the audio related contents to related ES_info loop in the Audio program.

When source functions use K_{XH0} instead of K_X/K_{XM}, the C_A2 and C_A fields in the PCP shall be set to zero and one, respectively.

V1SE 7.3.2 Modification to B.3.3.Treatment of the DTCP_descriptor by the sink device

This section replaces Section B.3.3 and describes the treatment of the DTCP_descriptor and DTCP_audio_descriptor when received by a sink device. When the function of the sink device is format cognizant and receives recognizable Embedded CCI other than the DTCP_descriptor and DTCP_audio_descriptor within an MPEG-TS, the alternative Embedded CCI shall take precedence over the information contained within the DTCP_descriptor or DTCP_audio_descriptor. Furthermore, the DTCP_descriptor and DTCP_audio_descriptor are only valid when they are inserted into the PMT. If a DTCP_descriptor or DTCP_audio_descriptor is found in another location, it shall be ignored.

When the only Embedded CCI detected is the DTCP_descriptor or DTCP_audio_descriptor, the DTCP_descriptor shall be regarded as the Embedded CCI described in Sections V1SE 4.15 and V1SE 4.16 except as otherwise noted, and the DTCP_audio_descriptor shall be regarded as the Embedded CCI described in Sections V1SE 4.22 and interpreted as follows:

- If a DTCP_descriptor or DTCP_audio_descriptor is found in an ES_info loop of the PMT, the Embedded CCI value contained in the descriptor should only be used as the CCI for the specific ES for which the DTCP_descriptor or DTCP_audio_descriptor is associated.
- When the only Embedded CCI detected in an ES_info loop of an Audio program is DTCP_descriptor, the DTCP_descriptor shall be regarded as the Embedded CCI described in only Section V1SE 4.16.
- If a DTCP_descriptor and DTCP_audio_descriptor is not found in the ES_info loop for a specific ES, but is instead found in the program_info loop, the Embedded CCI values contained within the DTCP_descriptor or DTCP_audio_descriptor shall be used as the CCI for that ES.
- A program in a stream shall be regarded as Copy-free if the stream contains multiple programs and none of Embedded CCI, DTCP_descriptor and DTCP_audio_descriptor is detected in the program and a DTCP_descriptor or DTCP_audio_descriptor is detected in another program on the same stream.

[19] as described in the definition of ISO/IEC 13818-1

V1SE 8 Modifications to Appendix C Limitation of the Number of Sink Devices receiving a Content Stream

In addition to Volume 1 Appendix C requirements the following requirements also apply:

- For clarity remotely connected sinks are included in the sink limitation count.
- All descriptions about Exchange Keys (K_X) and Session Exchange Keys (K_S) shall be interpreted as the descriptions about Exchange Keys (K_X), Session Exchange Keys (K_S) and Remote Exchange Keys (K_R). For example, when the source device expire all Exchange Keys (K_X), Session Exchange Keys (K_S) and Remote Exchange Keys (K_R), it shall reset the Sink Counter to zero and clear the list of registered Device IDs and ID_Us.

V1SE 9 Modifications to Appendix E Content Management Information (CMI)

V1SE 9.1 Modifications to E.1.2 General Rules for Source Device

When CMI capable source devices send usage rule by using CMI packet,

- Source devices may insert multiple CMI Descriptors in the single CMI packet. Source devices shall not insert plural CMI Descriptors with the same ID in the same CMI Field. When source devices send multiple CMI Descriptors in the single CMI packet, source devices shall arrange CMI Descriptors in ascending order of CMI Descriptor ID. Source devices shall not change the set of CMI Descriptors during transmitting content. Source devices shall not send CMI Descriptor when they do not support the CMI Descriptor.
- Source devices shall send a CMI packet before transmitting DTCP protected content using PCP2. Source devices may send one or more sequential separate PCP2 without a CMI Packet. Source devices may update data of the CMI Descriptor in the next CMI packet in the middle of content transmission using PCP2. Source devices may send the same CMI packet as the previous one even if the upstream does not change the usage rule.
- For RTP transfers, each RTP payload is encapsulated by a single PCP2 and a single CMI packet is transmitted in an RTP packet. When new CMI Descriptor is defined in the future and the size of transmitting CMI Field is larger than the maximum size of RTP payload, the CMI Field could be split into several RTP payloads.
- For HTTP transfers, source devices shall send CMI packet and PCP2 (associated to the CMI packet) in the same TCP connection. Source devices shall not send CMI packets in the HTTP transfers using PCP.

V1SE 9.2 Modifications to E.1.3 General Rules of Sink Devices

Sink devices shall apply the usage rule indicated by a CMI packet to the following PCP2(s) to which the CMI packet is associated until the next CMI packet.

V1SE 9.3 Modifications to E.3.3.1 CMI Descriptor 1 Format

Only when E-EMI is Mode C1, DTCP_CCI is No-more-copies(01_2), and the CC field is non-zero, is the CC field valid. In other conditions, the CC field is invalid.

V1SE 9.4 Modifications to E.3.3.3 Rules for Sink Devices

When Sink device receives a content stream with invalid CC field as specified in V1SE 9.3 they shall ignore CC field.

V1SE 9.5 Modifications to E.3.4.3 Rules for Sink Devices

When sink devices receive audio content, they shall ignore CC field in CMI Descriptor 2.

V1SE 10 Additional Requirements

V1SE 10.1 Authentication Capability Constraint

For DTCP-IP both source and sink devices shall only use Full Authentication.

V1SE 10.2 Internet Datagram Header Time To Live (TTL) Constraint

TTL is described in RFC791 and the following requirements only apply to IP datagrams that transport DTCP AKE commands. Transmitting devices shall set TTL value of such transmitted IP datagrams to a value no greater than 3 and correspondingly receiving devices shall discard such received IP datagrams which have a TTL value greater than 3 except for the following cases:

- Transmission and reception of IP datagrams required for RA-AKE (including CKC using K_R).
- Transmission and reception of IP datagrams for RA_MANAGEMENT subfunction data.
- Transmission and reception of IP datagrams for the DTCP-IP Move Protocol between a source device and sink device sharing a Remote Exchange Key.

V1SE 10.3 802.11 Constraint

DTCP devices with integrated 802.11 must ensure that either WEP or other such equivalent protection mechanism (e.g. WPA or WPA2) is engaged prior to exchanging DTCP AKE commands and protected content via such a network interface. For interoperability purposes devices must have at least WEP capabilities. Please note that this requirement to use WEP may be amended to require use of successor technologies as designated by DTLA.

V1SE 10.4 DTCP-IP Move Protocol

This section specifies a transaction based Move protocol[20] for a Move function using Mode C1 of E-EMI that uses a move specific Exchange key for each Move transaction. The transaction based Move protocol results in either the content being completely moved to the sink device (Success case) otherwise the content remain usable in the source with no usable content in the sink device (Cancel case). Source and sink devices that support the transaction based Move protocol shall support the requirements specified in this section.

The Move protocol consists of three parts; Move RTT-AKE, Move Transmission and Move Commitment. Each transaction based on the Move protocol (Move transaction) begins with Move request from a sink device and completes when the Move Commitment process completes or any one of these processes are canceled or aborted.

An unique Exchange key (K_{XM}) is generated specifically for each Move transaction during Move RTT-AKE. K_{XM} is used to calculate the Content key (K_C) used to encrypt the moved content. Content received by the sink device remains unusable until the successful completion of the Move Commitment phase of the Move transaction. Upon successful completion of the Move Commitment phase the moved content in the source device is made unusable and the moved content in the sink device is made usable.

Both source and sink devices can cancel a Move transaction any time before starting the Move Commitment process.

For transaction based Move function the Session Exchange Key shall not be used.

[20] Without using this Move protocol, move of content based on Exchange key (K_X) may be performed as specified in V1SE 4.27.

V1SE 10.4.1 Move RTT-AKE

Source devices generate an Exchange key (K_{XM}) specifically for the Move transaction and to calculate the Content key (K_C) used to encrypt the content to be moved during the Move transaction.

Move RTT-AKE is used to exchange K_{XM} and associated protocol flow is shown in following figure.

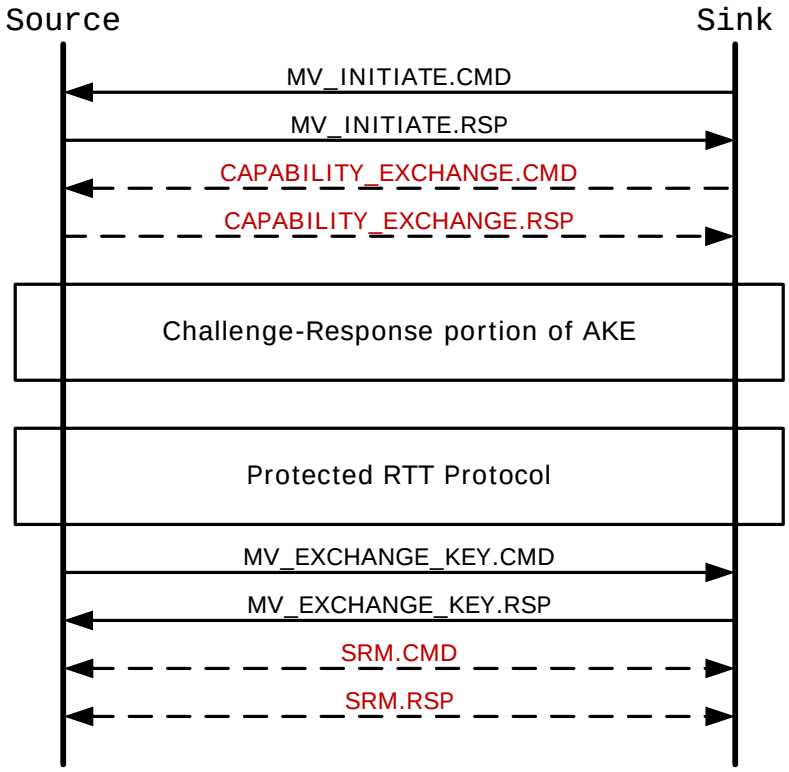


Figure 6 Move RTT-AKE Protocol Flow

1. Sink device initiates the Move RTT-AKE protocol by sending MV_INITIATE command. If source device can perform the DTCP-IP Move protocol, the source device returns response as accepted.
2. If sink device needs to exchange capabilities, the sink device may send CAPABILITY_EXCHANGE command at this point.
3. Challenge-Response portion of AKE and Protected RTT protocol (see section V1SE 10.5.1) are executed subsequently to share Authentication key for Move (HK_{AUTH}). In the Challenge-Response portion of AKE, sink device sets only bit 3 (K_X) to one in the exchange_key field in the CHALLENGE command (even in the Remote Access) and source device performs the Sink counting specified in Appendix C of Volume 1 specification. Source devices may skip Protected RTT Protocol when sink device is on its RTT Registry as specified in V1SE 10.5.2 or if the Device ID or ID_U (if common key device) obtained during "Challenge-Response portion of AKE" exists in the RAC Registry for remote access.
4. Source device generates a Move Exchange key (K_{XM}) and sends it to the sink device. (See the following section for detail)

V1SE 10.4.1.1 *Establishing Move Exchange Key*

Source device establishes the Move Exchange key (K_{XM}) and sends it to sink device using the following procedure:

1. The source device shall assign a random value for the Move Exchange key (K_{XM}) (using RNG_F) being established. The source device assigns K_{XM}_label to this K_{XM}.
2. The source device then scrambles the key K_{XM} using HK_{AUTH} (calculated using K_{AUTH}) resulting in K_{SXM} according to the function described in the DTCP Specification available under license from the DTLA.
3. The source device sends K_{SXM} and K_{XM}_label to the sink device.
4. The sink device descrambles the K_{SXM} using HK'_{AUTH} (calculated using K'_{AUTH}) to determine the shared K_{XM} according to the function described in the DTCP Specification available under license from the DTLA.

Source devices use the value of K_{XM}_label to identify the corresponding Move transaction in the Move Transmission and Move Commitment processes. Source devices shall not use the value of K_{XM}_label assigned to the Move transaction(s) that have not yet completed.

Source and sink devices shall manage K_{XM} and K_{XM}_label as follows:

- K_{XM} shall be managed independent of the other Exchange Keys in terms of generation and expiration. K_{XM}_label may have the same value as the exchange_key_label of the other Exchange Keys.
- K_{XM} and K_{XM}_label can only be used in the corresponding Move transaction and shall not be used for other purposes.
- K_{XM} and K_{XM}_label shall be expired when the corresponding Move transaction completes regardless of result.
- It is mandatory that the source device expires a K_{XM} within 2 hours after Move Transmission using the K_{XM} has ceased.
- It is mandatory that the sink device expires a K_{XM} within 2 hours of continuous non-use of that K_{XM} for decryption.
- Source and sink devices must expire their K_{XM} when they detect themselves being disconnected from all mediums. For wireless mediums this means when device detects that it is not connected to an access point or it is not directly connected to another device.
- When K_{XM} is expired the K_{XM}_label shall also be expired except when the K_{XM}_label is stored for resumption of Move Commitment. (See section V1SE 10.4.3.1)

Note that the source device shall not reset the Sink Counter when K_{XM} is expired except for the case that the source device shares neither Exchange key (K_X, K_S, or K_R) nor Move Exchange key other than the K_{XM} with any sink device.

V1SE 10.4.2 Move Transmission

The Move Transmission process starts upon the completion of the Move RTT-AKE and is part of the Move transaction where the moved content is encrypted using the Content key K_C, calculated using K_{XM} instead of K_X and using Mode C1 (Move Audiovisual) of E-EMI in Move Transmission (See section V1SE 4.23). The source device shall set the value of K_{XM}_label to exchange_key_label field in PCP/PCP2.

Note for the following cases:

1. When the CMI is being used, K_{XM} shall be used instead of K_X to calculate AK_C (see V1SE 4.4).
2. When the CMI is not used and content includes the DTCP_descriptor with DOT-asserted, K_{XM} shall be used instead of K_X to calculate K_{XH0} (see B.3.1) and that K_{XH0} shall be used instead of K_{XM} to calculate K_C.
3. ~~2.~~ When CMI is not used and PCP-UR with DOT-asserted is used, K_{XM} shall be used instead of K_X to calculate K_{XH0} (see B.3.1 of Volume 1 Specification) and K_{XH0} shall be used instead of K_{XM} to calculate K_C.

Source devices shall not encrypt the same part[21] of content more than once using K_{XM} during a Move transaction. Source devices shall prevent content from plural transmission for move unless otherwise noted.

Sink devices shall keep the content received during Move Transmission unusable until successful completion of the Move Commitment process, except for the use of the receiving content as if it has Mode C0 of E-EMI.

When HTTP is used for the Move Transmission, source device and sink devices must not initiate another HTTP transfer[22] for the Move Transmission before completing an HTTP transfer for the Move Transmission in a single Move transaction. Refer section V1SE 12.4 for recommended HTTP header field.

In the content key confirmation procedure during Move Transmission, K_{XM} shall be used instead of K_X to calculate a MAC value by both source device and sink device (See section V1SE 10.6: Content Key Confirmation). Source devices shall manage the value of N_C in conjunction with the value of K_{XM}_label (Note that there is only one N_C value for a K_{XM}_label at a time). Source devices shall compare received N_CT with N_C corresponding to received K_{XM}_label.

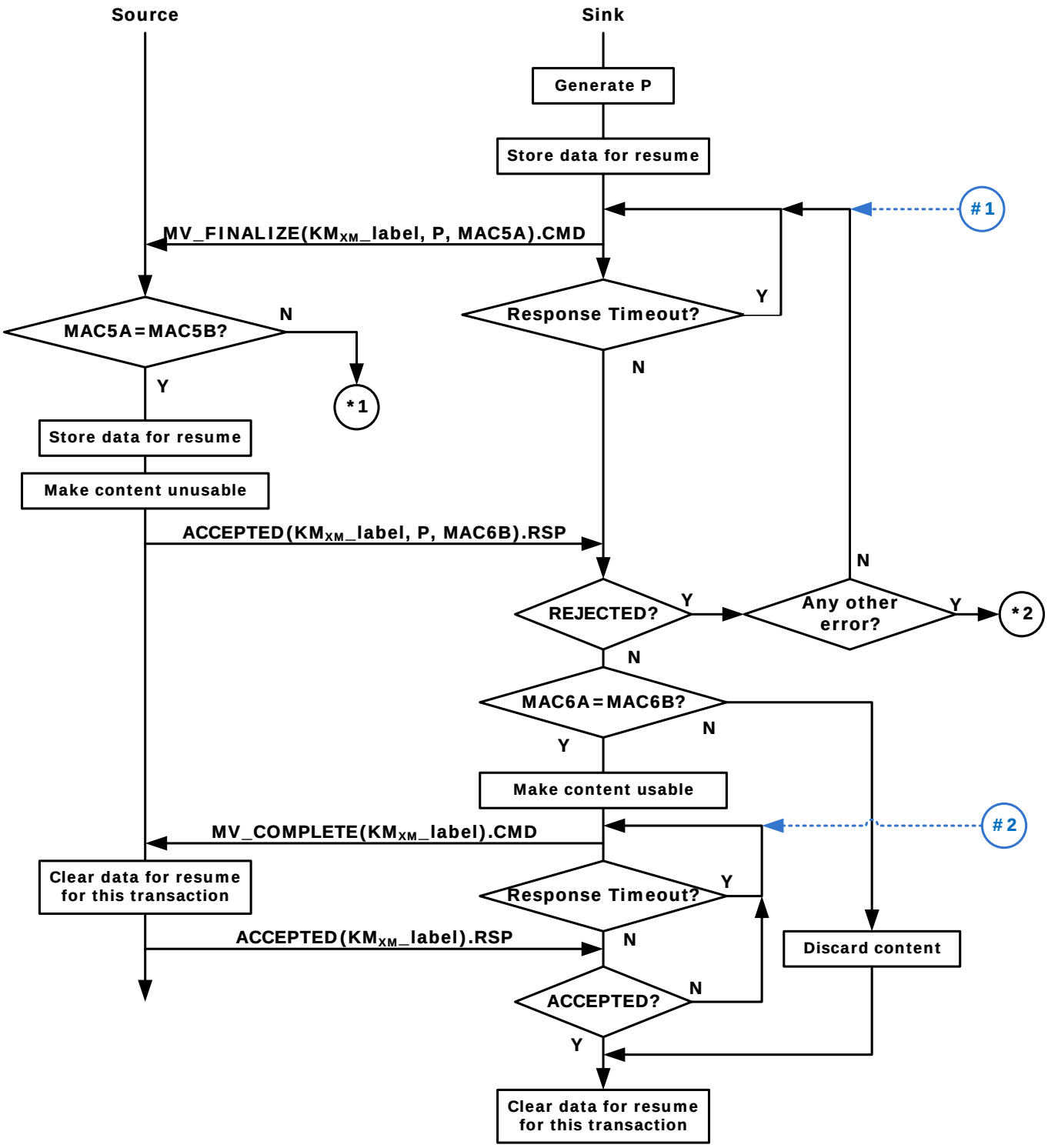
V1SE 10.4.3 Move Commitment

Sink devices initiate the Move Commitment process when Move Transmission has completed.

Sink device can make received content usable only upon the successful completion of the Move Commitment process. The following figure depicts the Move Commitment protocol flow.

[21] The content may be retransmitted in transport protocol (ex. TCP).

[22] Source devices may not be capable of supporting Move transaction via multiple HTTP transfers in a single Move transaction.



***1** Source device is recommended to return REJECTED.RSP with "Any other error" status and keep waiting for MV_FINALIZE.CMD. However, it may cancel the Move transaction if the content has not yet been made unusable, then it should return REJECTED.RSP with "Any other error" and clear resume data for this transaction (if stored).

***2** Sink device is recommend to resend MV_FINALIZE.CMD after reconfirming IP address of source device with which K_{XM} has been exchanged. However, it aborts the Move Commitment process if result is the same. When it aborts, it should clear resume-data for this transaction.

Figure 7 Move Commitment Protocol Flow

SHA-1 is used to construct following MAC values that are exchanged during the Move Commitment protocol to ensure that the source device and the sink device share K_{XM}.

- MAC5A = MAC5B = $[SHA\text{-}1(MJ+P)]_{msb80}$
- MAC6A = MAC6B = $[SHA\text{-}1(MJ+P)]_{lsb80}$

Where MJ is 160 bits and equal to SHA-1(K_{XM} || K_{XM}), and K_{XM} corresponds to K_{XM}_label in the MV_FINALIZE command. P is a 64 bits random number (generated by RNG_F). The "+" is used in the above formula to mean mod 2^{160} addition.

Source devices compute MAC5B and compares it to MAC5A when MV_FINALIZE command is received. If not equal, the source device returns REJECTED response with "Any other error" status; else if equal, it shall make content transmitted in the Move Transmission unusable and returns ACCEPTED response to the sink device.

Sink devices compute MAC6A and compares it to MAC6B when ACCEPTED response is received. If not equal, the sink device completes the Move transaction and discards any received content; else if equal, it makes content received in the Move Transmission usable and sends the MV_COMPLETE command to the source device.

When the sink device detects a timeout before receiving the ACCEPTED response to the MV_FINALIZE command, it should resend the MV_FINALIZE command unless REJECTED response with "Any other error" status is received from the source device with which K_{XM} was exchanged.

Source device completes the Move transaction after sending the ACCEPTED response when the MV_COMPLETE command is received. Sink device completes the Move transaction when the ACCEPTED response is received.

When sink devices detect a timeout before receiving the ACCEPTED response to the MV_COMPLETE command, it should resend the MV_COMPLETE command not to leave data for the Move Commitment process in sink device (and source device).

V1SE 10.4.3.1 *Resumption of Move Commitment*

There is a brief period in the Move Commitment process where Moved content is marked unusable in both the source and sink device such that if an interruption (e.g. loss of TCP connection) were to occur at this point in the process it would result in loss of moved content. To avoid this, it is recommended that both source and sink device store[23] required data[24] to complete Move Commitment protocol into NVRAM and perform the following resume procedure. The data is stored at the beginning and cleared at the end of the Move Commitment protocol as shown in V1SE 10.4.3.

In case of a broken AKE TCP connection, the TCP connection must first be reestablished between the affected source and sink device. When sink devices cannot get a DTCP socket without notification from source device (e.g. content-push type Moves), the source device should transmit HTTP POST request[25] with DTCP socket in the POST header to the sink device.

[23] At least the device should keep the stored data while the device is power-on.

[24] For example, parameters required in Move Commitment and information to discover device and moved content. Note that to keep this information unchanged is essential for resume of Move Commitment (e.g. UPnP AV CDS Object ID).

[25] To the same destination as Move Transmission without message-body.

The sink device should execute the procedure shown below after communication with the source device is reestablished and where #1 and #2 are the entry points specified in Figure 7.

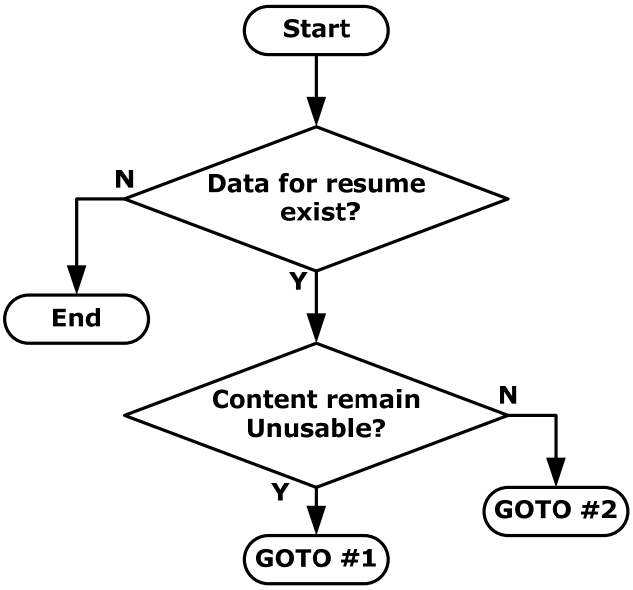


Figure 8 Resume procedure for sink device

The source device should execute the procedure shown below based on the K_{XM}_label specified in the MV_FINALIZE command or the MV_COMPLETE command when one of these two commands is received.

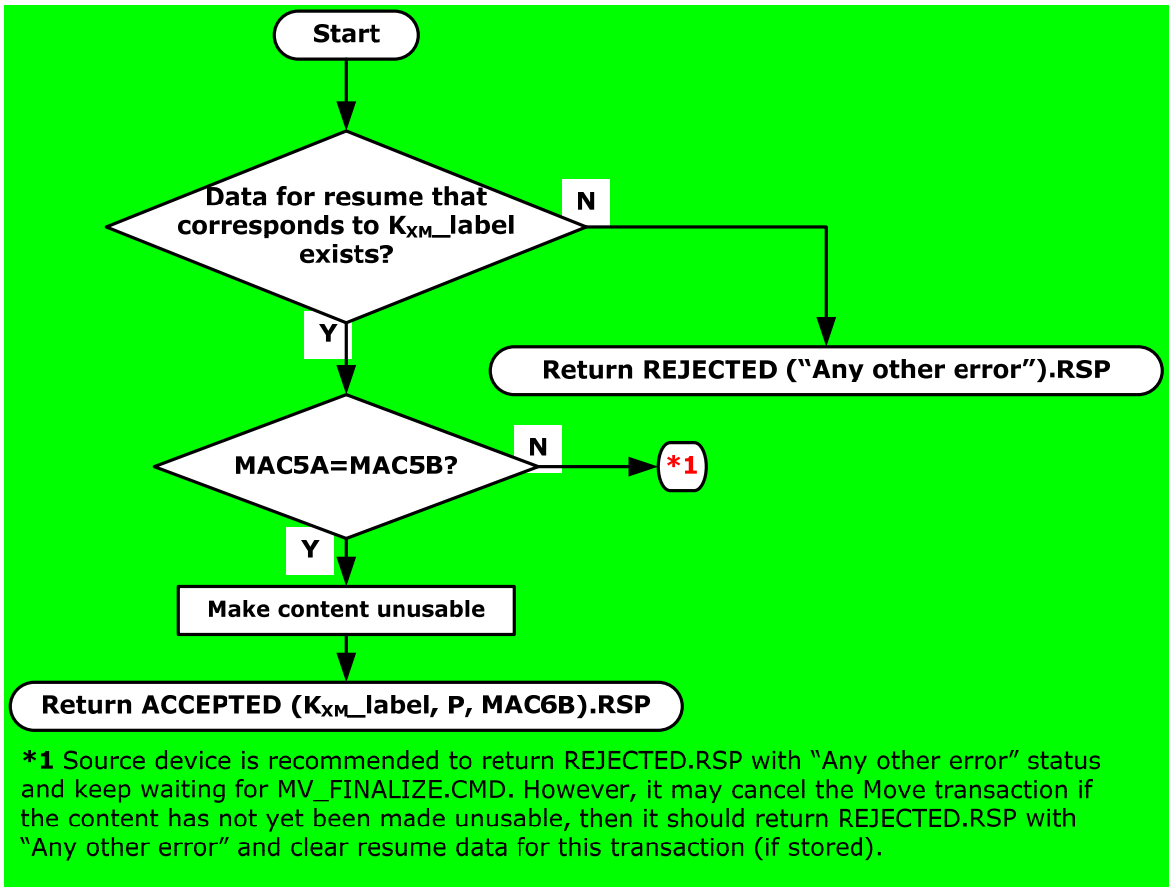


Figure 9 Resume procedure for source device when MV_FINALIZE is received

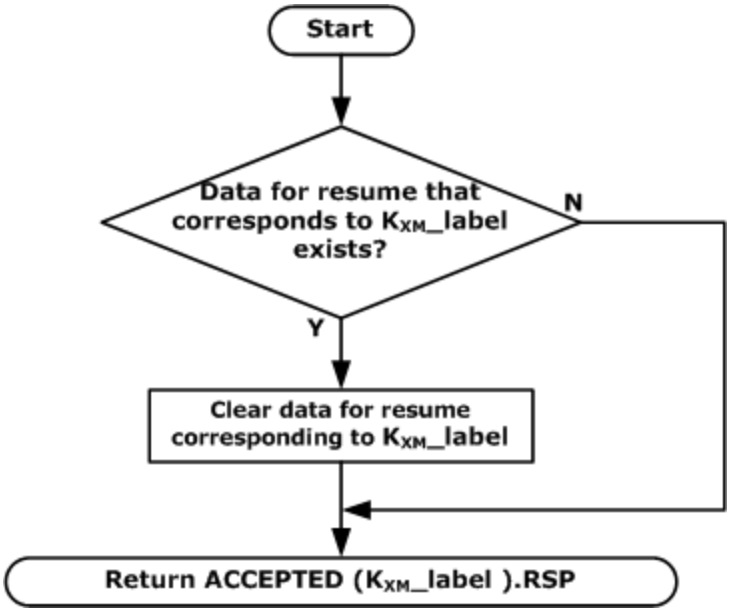


Figure 10 Resume procedure for source device when MV_COMPLETE is received

The source device should return the ACCEPTED response to the MV_COMPLETE command even when it has already cleared data for resume.

V1SE 10.4.4 Cancel of Move transaction

Source devices can cancel the Move transaction without disabling its content before issuing the first ACCEPTED response to the MV_FINALIZE command. Sink device can cancel Move transaction as if it has received no content before issuing the first MV_FINALIZE command.

Sink devices which cancel a Move transaction shall discard content received during the Move Transmission in the Move transaction.

During the Move RTT-AKE process, the device desiring to cancel the Move transaction should send the AKE_CANCEL command.

During the Move Transmission process, the device desiring to cancel the Move transaction should send the MV_CANCEL command. It is recommended that source and sink devices maintain the AKE TCP connection until completion of the MV_CANCEL command from source device.

During the Move Commitment process, source device should return the REJECTED response with “Any other error” status to the MV_FINALIZE command when it cancels the Move transaction. Source device shall not return the REJECTED response with “Any other error” status to the MV_FINALIZE command to cancel[26] the Move transaction if it has already issued the ACCEPTED response for the MV_FINALIZE command of the Move transaction. Source and sink devices shall clear data stored for resume corresponds to the Move transaction being canceled.

[26] Source devices that do not support the resumption of the Move Commitment may return the REJECTED response with “Any other error” status.

V1SE 10.5 Additional Localization via RTT

Source and sink devices must implement Additional Localization as specified in this section. Note that RA-AKE is not required to implement RTT.

Source devices with Additional Localization (AL) when conducting an AKE with a Sink device with AL must perform a RTT test if the sink device's Device ID is not on the source device's RTT registry.

Source devices will add a Sink device's Device ID to the Source device's RTT registry, set the content transmission counter for the sink device to 40 hours, and provide an exchange key only if the source device measures a RTT value of 7 milliseconds or less during RTT test.

Source devices when transmitting content will update content transmission counters of all RTT registered sink devices and are required to remove the Device ID of a sink device from the RTT registry after counting 40 hours of content transmission.

Background RTT testing is not a required capability. If background RTT testing is supported, the source device will add the sink device's Device ID to the RTT registry if not registered and set content transmission counter to 40 hours only if the source device measures a RTT value of 7 milliseconds or less during RTT test.

When RESPONSE2 subfunction is received, ID_U shall be used instead of Device ID in above processes.

V1SE 10.5.1 Protected RTT Protocol

DTCP-IP's protected RTT protocol is described in Figure 11 and is used in RTT-AKE and Background RTT check procedures. The RTT protocol is executed after the Challenge-Response portion of the AKE is completed. SHA-1 is used to construct the following messages that are exchanged during RTT testing protocol to ensure that source and sink which completed Challenge-Response portion of AKE are only ones involved in RTT testing.

- MAC1A = MAC1B = $[\text{SHA-1}(MK+N)]_{msb80}$
- MAC2A = MAC2B = $[\text{SHA-1}(MK+N)]_{lsb80}$
- OKMSG = $[\text{SHA-1}(MK+N+1)]_{msb80}$
  Where MK is 160 bits and equal to SHA-1(Kauth||Kauth), N is 16 bit number that ranges from 0 to 1023, and "+" used in RTT Protocol means mod 2^{160} addition.

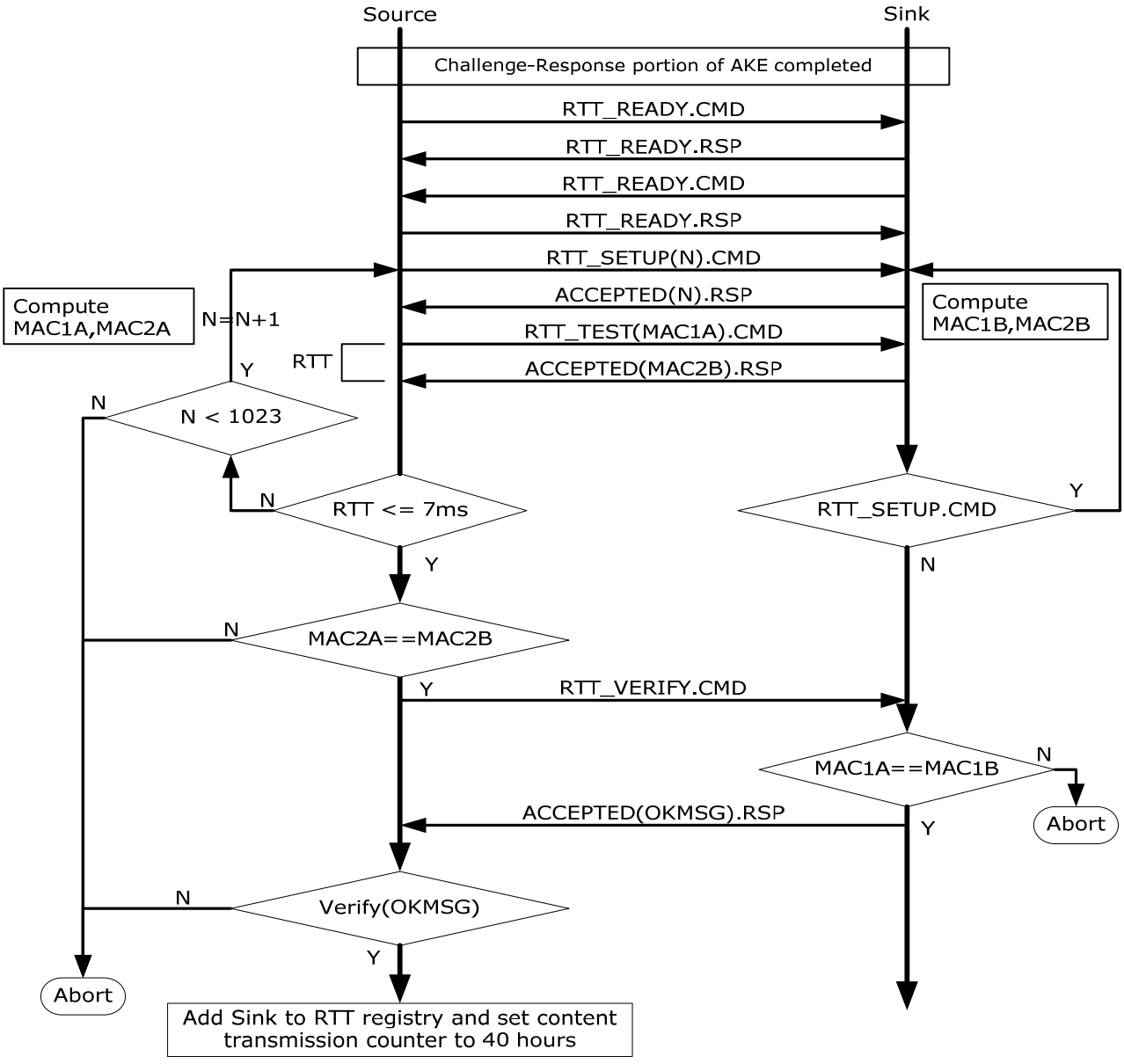


Figure 11 RTT Protocol Diagram

The RTT_READY command is used to indicate that authentication computation is complete and that source and sink devices are ready to execute the RTT test procedure.

The RTT procedure begins by first establishing value of N using the RTT_SETUP command. N is initially set to zero and can range from 0 to 1023 as maximum permitted RTT trials per AKE is 1024.

After preparation of MAC values corresponding to N, source device will then measure RTT which is the time interval starting after source transmits RTT_TEST command and terminates upon reception of RTT_TEST accepted response.

If the RTT is greater than 7 milliseconds and the value of N is less than 1023 the source will repeat RTT procedure by incrementing N by 1 and reissue RTT_SETUP and RTT_TEST commands.

If the measured RTT is less than or equal to 7 milliseconds:

The source device compares most recently computed MAC2A to most recently received MAC2B and if not equal the source device aborts RTT procedure else if equal it sends RTT_VERIFY command to sink device.

The sink device will after receipt of RTT_VERIFY command compare the most recently received MAC1A and most recently computed MAC1B and if not equal aborts RTT procedure else if equal it will send OKMSG in RTT_VERIFY accepted response.

The source device will verify OKMSG and if it is not correct the source device aborts RTT procedure else it will add sink device's Device ID to RTT registry and set content transmission counter to 40 hours. When RESPONSE2 subfunction is received, ID_U shall be used instead of Device ID in above process.

If RTT procedure is aborted the source shall not provide an exchange key.

V1SE 10.5.2 RTT-AKE

The RTT-AKE procedure starts exactly the same as normal AKE but source and sink devices that have DTCP certificates with AL flag set to one must check AL flag value of other device and if the AL flag value is also set to one then:

The sink device after completing Challenge-Response portion of AKE will wait and the sink device will abort if it receives any other command than the RTT_READY command, EXCHANGE_KEY command, or AKE_CANCEL command.

The source device then examines the RTT registry and if the sink device's Device ID is on its RTT registry, the source device proceeds to exchange key portion of AKE otherwise the source device initiates a RTT test procedure and if during test it obtains a RTT measurement of 7 milliseconds or less it will add the sink device's Device ID to its RTT registry, set content transmission counter to 40 hours, and then proceed to exchange key portion of AKE. When RESPONSE2 subfunction is received, ID_U shall be used instead of Device ID in above process.

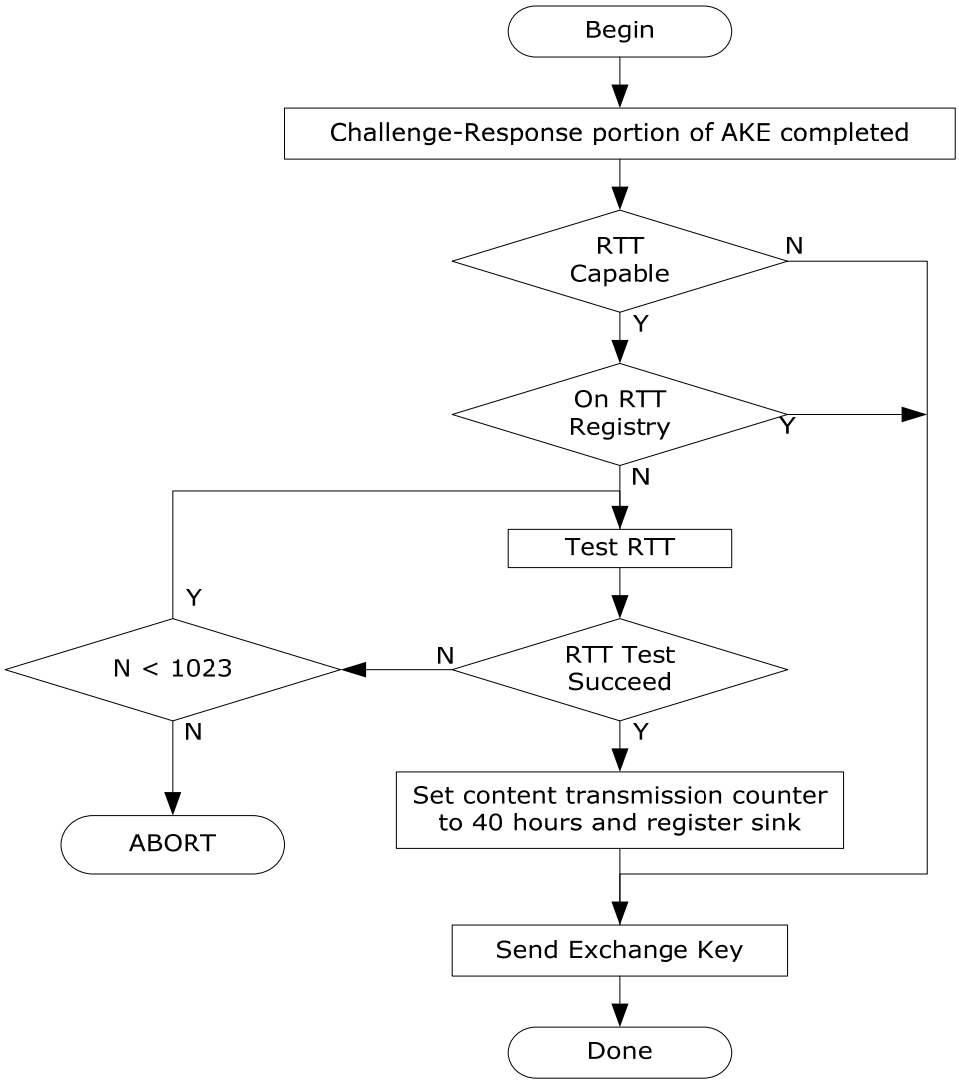


Figure 12 AKE-RTT Informative Flow Diagrams

V1SE 10.5.3 Background RTT Check

The Background RTT check procedure permits either the source or sink device to initiate an RTT background check which is only used to add the sink device to the source device's RTT registry if the sink device's ID is not already on RTT registry or if the sink device which is already on the source device's RTT registry, sets the content transmission counter to 40 hours. For the case of a Background RTT check, source devices shall not transmit an exchange key.

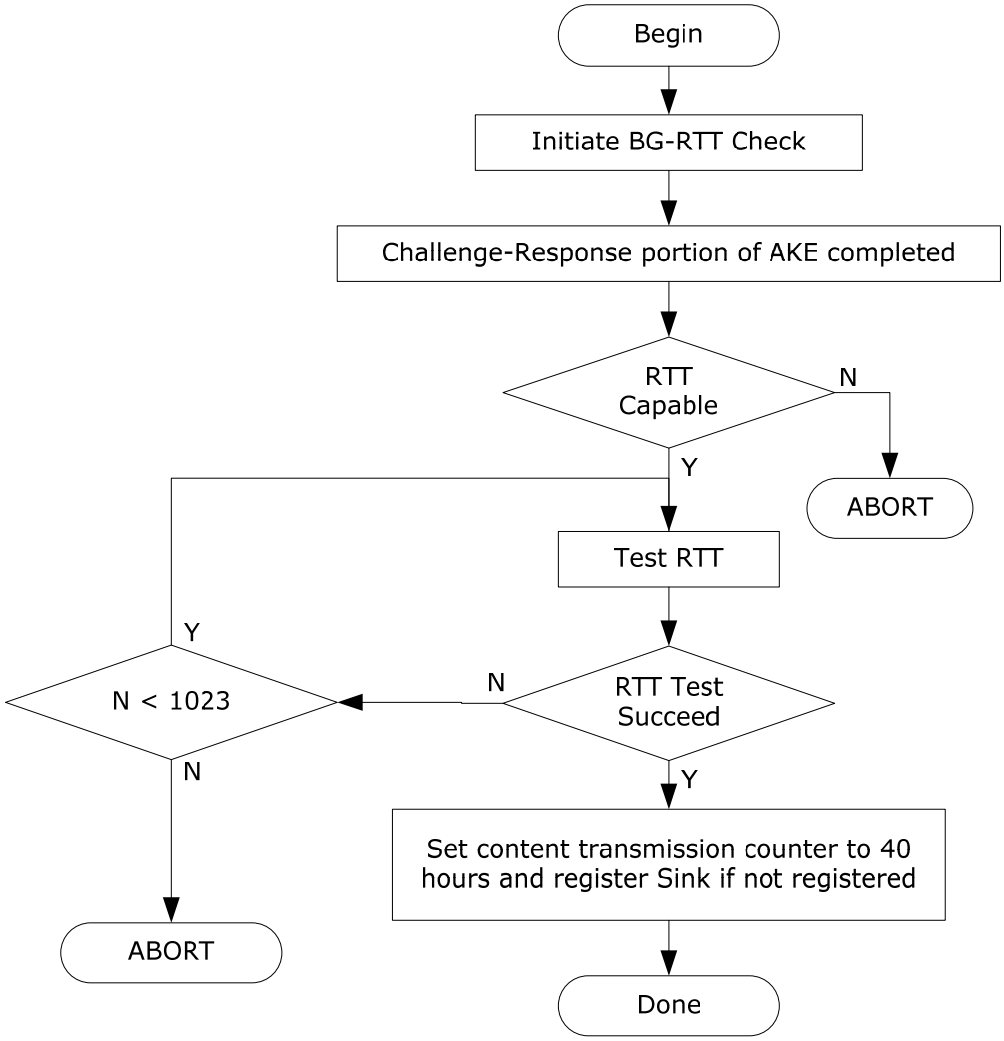


Figure 13 Background RTT Check Informative Flow Diagram

V1SE 10.6 Content Key Confirmation

For interoperability the content key confirmation function is limited to only those source and sink devices whose AL flag has a value of one. The sink device uses the CONT_KEY_CONF subfunction to confirm that the content key via the associated N_C is current.

Sink devices must monitor and confirm the N_C value of the most recently received PCP/PCP2 containing encrypted content for each content stream and then periodically reconfirm subsequent N_C(s) at least every 2 minutes. Periodic confirmation of N_C can be avoided if after initial confirmation, the sink monitors and confirms that subsequent N_C values are monotonically increasing contiguous values. Sink devices that have confirmed that the associated source device supports PCP-UR may use SN_C as a substitute for N_C.

Per content stream, sink devices after an initial non-confirmation of a N_C have one minute to repeatedly attempt to confirm a subsequent N_C values before they must terminate decryption for that content stream.

Sink devices may restart decryption upon confirmation of any N_C after a N_C non-confirmation event.

The content key confirmation procedure requires the sink device to send the N_C value under test (N_CT) to the source device. Upon receipt the source device checks the received N_CT against its current N_C values and if any are within the range N_CT to N_CT+5 then it confirms that N_CT is valid. Note that source devices which support PCP-UR shall use only the least significant 48 bits of both N_C and N_CT for this check since upper 16 bits are used for PCP-UR. The confirmation procedure is depicted in following figure.

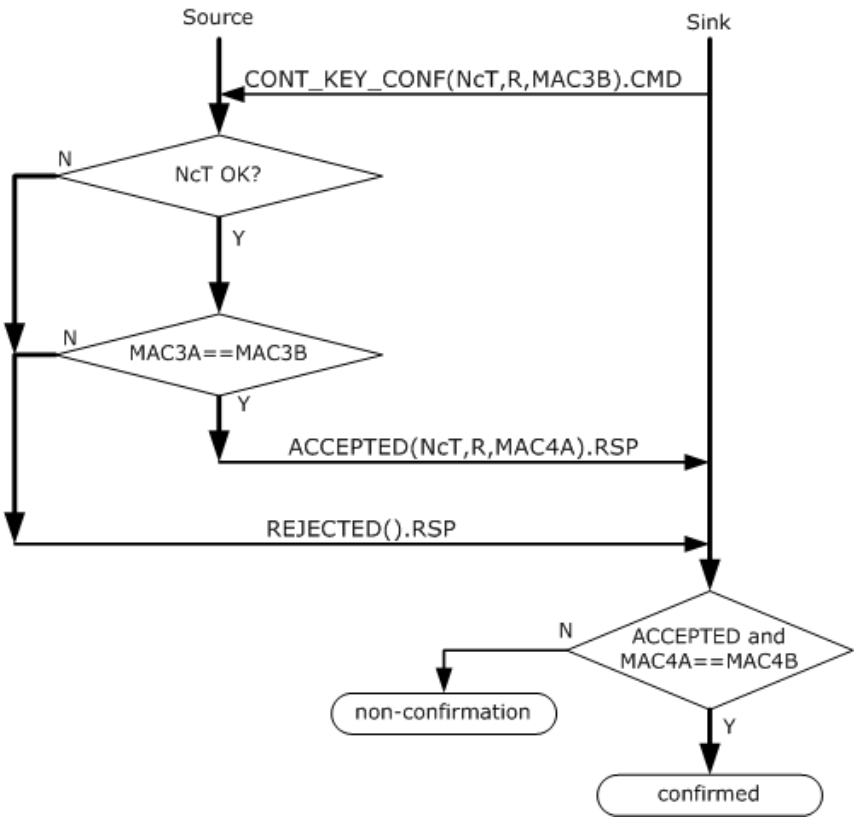


Figure 14 Content Key Confirmation Procedure

Where:

MX=SHA-1(Kx||Kx),
R is 64 bits, its initial value is a random number and is incremented by 1 mod 2^{64} for subsequent trials.
MAC3A = MAC3B = [SHA-1(MX+NcT+R)]msb80
MAC4A = MAC4B = [SHA-1(MX+NcT+R)]lsb80
"+" used in the above formulas means mod 2^{160} addition

Note that when Session Exchange Key (K_S) is used for the content transmission, K_S shall be used instead of K_X and when Remote Exchange Key (K_R) is used for content transmission, K_R shall be used instead of K_X.

V1SE 10.7 Remote Access

Remote access refers to the case where Remote Access capable source devices permit sink devices to request and connect to the source device without executing Additional Localization procedures if the sink device is on the source device's Remote Sink Registry.

Remote Access capable source device requirements:

- Source devices must maintain a Remote Sink Registry.
- Source devices will add only those sink devices that successfully pass the remote registration protocol to the Remote Sink Registry by recording the Sink-ID which is either the Device ID of the sink device's Certificate or ID_U of the Sink device.
  - For sink devices with common keying material, the source device shall~~will~~ record the ID_U instead of the Device ID of the sink device.
- The Remote Sink Registry is limited to 20 devices.
  - Record(s) in the Remote Sink Registry may be removed as needed.
- Source devices will permit only the prescribed number[27] of remotely connected sink device(s) at any time.
- Source devices will only permit AKE without RTT testing and TTL checking to proceed if the sink device's Sink-ID contained in the source device's Remote Sink Registry.

V1SE 10.7.1 Remote Sink Registration

Source devices will add a sink device to their Remote Sink Registry only if the connected sink device successfully passes the Remote Sink Registration protocol as described in the following figure:

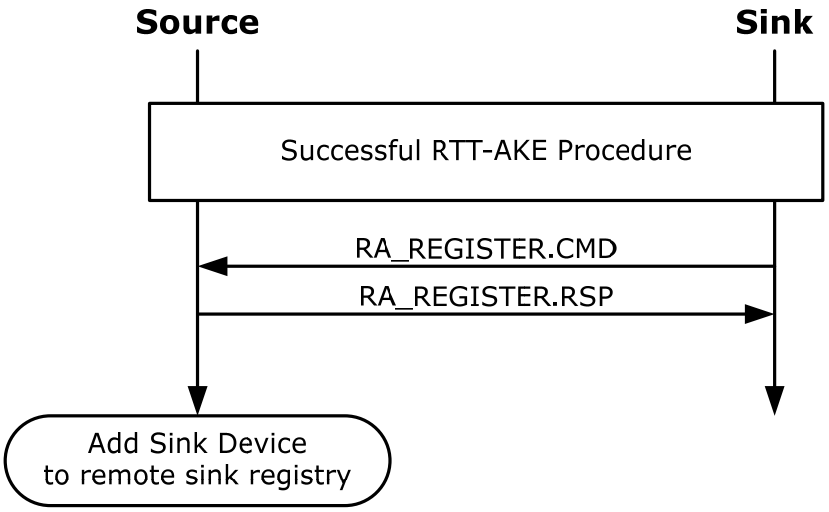


Figure 15 Remote Sink Registration Procedure

The Remote Sink Registration procedure is as follows:

1. After completing RTT-AKE procedure successfully, sink device sends its Sink-ID (Device ID or ID_U) using RA_REGISTER.CMD.
2. Source device checks that the Sink-ID is the same as the Device ID or ID_U (if common key device) received in the RTT-AKE procedure completed immediately before.
3. Source device checks whether the Sink-ID has already been stored in its Remote Sink Registry. Skip the following steps 4, and 5 if the Sink-ID is already stored.
4. If the Sink-ID has not been registered, source device checks whether its Remote Sink Registry is not yet full.
5. If checks in both step 2 and 4 are passed, source device adds the Sink-ID to its Remote Sink Registry.
6. Source device returns RA_REGISTER.RSP with the result of registration.

[27] Refer to Adopter Agreement

V1SE 10.7.1.1 *Mutual Registration*

Between devices that have capabilities of both source function (RA-SRC) and sink function (RA-SINK) capability of remote access, mutual registration is possible in a single Remote Sink Registration procedure as long as the source function can add another Sink-ID to their Remote Sink Registries. For avoidance of doubt, source device may not request mutual registration when it is possible.

RTT-AKE Source

Device A Device ID-A	
RA-SRC-A	RA-SINK-A

RTT-AKE Sink

Device B Device ID-B or ID_U-B	
RA-SRC-B	RA-SINK-B

In mutual registration, device A's RA-SRC-A registers device B's Sink-ID and the device B's RA-SRC-B registers device A's Sink-ID as Remote access sink in parallel. It is executed when device B's RA-SRC-B declares that the mutual registration is acceptable in RA_REGISTER.CMD and device A's RA-SINK-A requests the mutual registration in RA_REGISTER.RSP that also includes device A's Sink-ID. Note that device B shall not declare that mutual registration is acceptable to multiple devices in parallel, and device A with common device certificate shall not request the mutual registration. In the case of mutual registration, the following additional steps are continued after step 5 in the above Remote Sink Registration procedure:

6. Source device returns reject response and aborts the mutual registration if the check in step 2 or 4 fails, otherwise it returns accepted response with the source device's Sink-ID (Device ID) and flag to request mutual registration when sink device declares that mutual registration is acceptable.
7. Sink device checks that the source device's Sink-ID is the same as the Device ID received in the RTT-AKE procedure that immediately preceded remote registration procedure.
8. Sink device checks whether or not the source device's Sink-ID has already been stored in its Remote Sink Registry, and adds the source device's Sink-ID to its Remote Sink Registry.

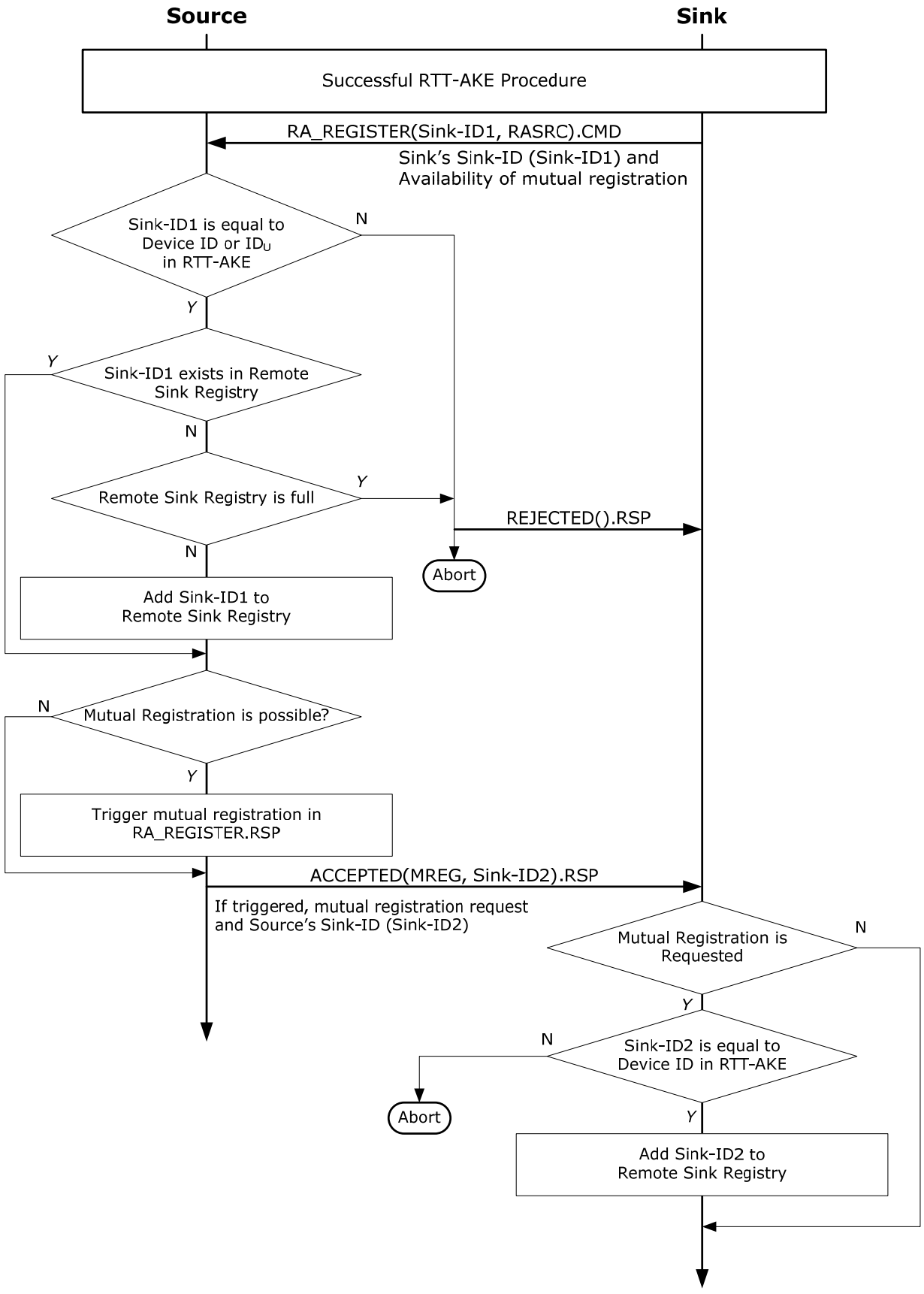


Figure 16 Mutual Remote Sink Registration Procedure

V1SE 10.7.2 Remote Access AKE (RA-AKE)

Remote access permits sink devices that are listed on the Remote Sink Registry to establish a remote connection to a specific source device if the source device has an unused remote access connection. A Remote Access Connection Registry (RAC Registry) is used to manage each RAC record which consists of Sink-ID, corresponding Remote Exchange Key (K_R), and exchange key label. The following figure shows the RA-AKE procedure to establish the remote access connection.

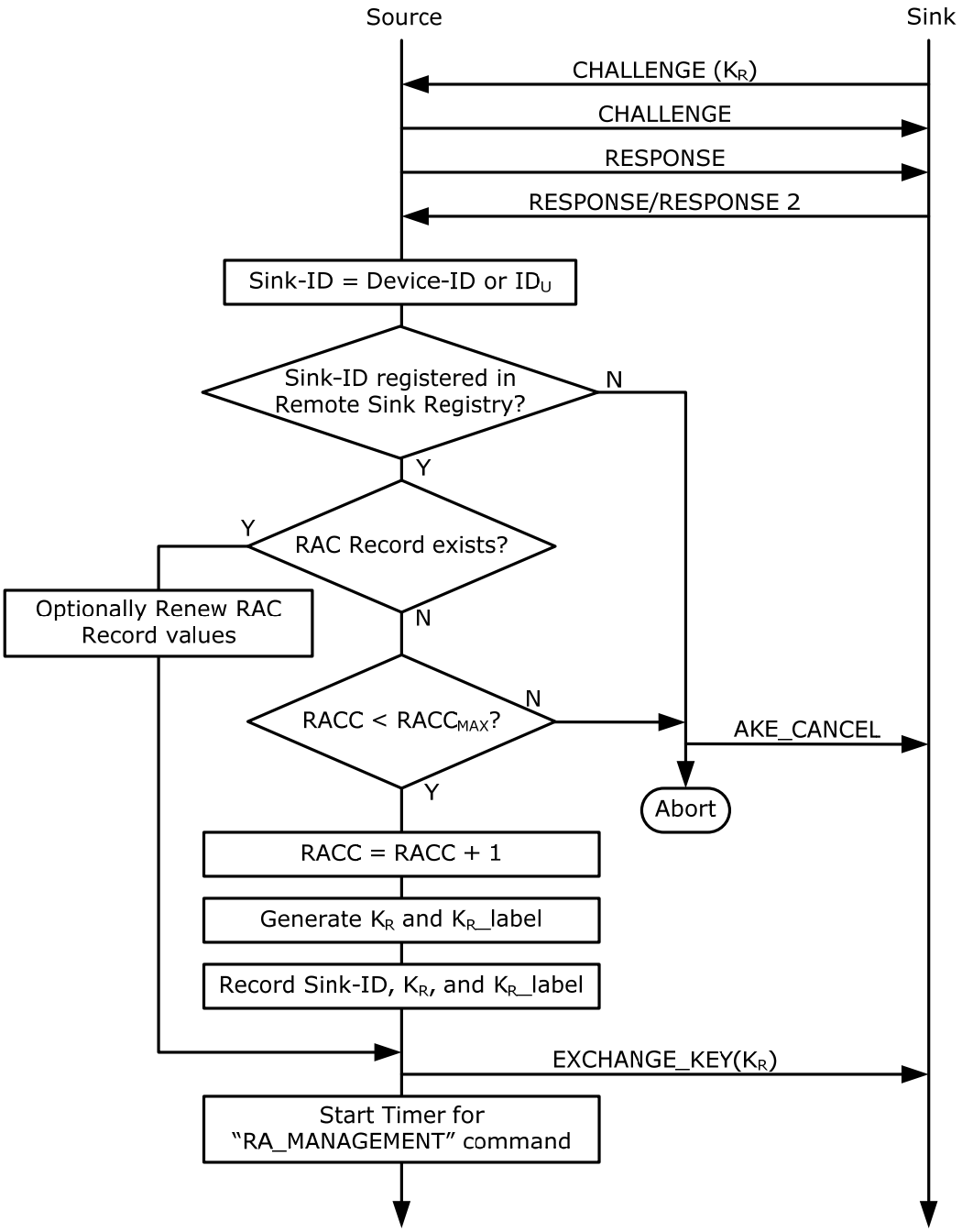


Figure 17 RA-AKE Procedure

RA-AKE procedure is as follows:

1. Sink device sends CHALLENGE with the exchange_key field in which the bit for K_R (Remote Exchange Key) is set. If the bit for K_R is not set, source devices shall abort the RA-AKE procedure. Note, that AKE procedure other than RA-AKE may be continued.
2. Source and sink devices execute the Challenge-Response portion of Full Authentication.
3. Source devices check whether the Sink-ID of the sink device is listed in its Remote Sink Registry. If Sink-ID is not listed, then it sends the AKE_CANCEL and aborts the RA-AKE procedure.
4. Source devices check the RAC Registry to determine if a RAC record exists for the given Sink-ID. If a RAC record exists, then the source device uses the K_R and corresponding exchange_key_label in the RAC record and proceeds to Step 8. Source devices may renew the values of K_R and exchange_key_label of the RAC record before going to Step 8 if the source device is not transmitting content using K_R.
5. If a RAC record does not exist for the given Sink-ID, the source device will then check the RACC Value. If RACC is NOT less than $RACC_{MAX}$, the source device must send AKE_CANCEL and abort the RA-AKE procedure. RACC is the counter for remote access connections which is initialized to zero when there are no remote access connection.
6. Source devices then increments RACC by one.
7. Source devices then generates K_R and corresponding exchange_key_label for the K_R, and put them in a RAC record along with the Sink-ID.
8. Source devices then send the Remote Exchange Key K_R and corresponding exchange_key_label associated to the Sink-ID to the sink device.
9. Source devices that support RA_MANAGEMENT then start a K_R keep-alive timer to maintain the K_R and retain the K_R for at least one minute.
10. SRM transmission follows if an update between source device and sink device is indicated.

When source devices expire a K_R, the source device erase the associated RAC record that contains K_R, and decrements the RACC by one.

V1SE 11 Additional Commands and Sequences

These additions defined for DTCP-IP are described in the DTCP specification available under license from the DTLA.

V1SE 12 Recommendations

V1SE 12.1 Recommended MIME type for DTCP protected content

The DTCP application media type is as follows:

application/x-dtcp1; CONTENTFORMAT=<mimetype>

Where **CONTENTFORMAT**, is the standard content media type that is protected by DTCP.

In addition, information identifying a DTCP Socket may be included as follows:

application/x-dtcp1; DTCP1HOST=<host>; DTCP1PORT=<port>; CONTENTFORMAT=<mimetype>

Refer to V1SE 12.2.1 for description of DTCP1HOST and DTCP1PORT.

In the case of content applicable to Remote access, information may be added as follows:

application/x-dtcp1; DTCP1HOST=<host>; DTCP1PORT=<port>; DTCP1RAPORT=<port>; CONTENTFORMAT=<mimetype>

Refer to V1SE 12.2.1 for description of DTCP1HOST, DTCP1PORT and DTCP1RAPORT.

Content type of HTTP response / request is set to DTCP application media type.

V1SE 12.2 Identification of DTCP Sockets

DTCP uses a TCP port to support various command and control protocols (e.g. AKE, Exchange Keys, SRM) and either a TCP or UDP for content transport. This section details recommended practices for identifying DTCP Sockets.

V1SE 12.2.1 URI Recommended Format

This following information is inserted into the query string portion of URI and is used to communicate the source's content and DTCP Socket to the sink. The source obtains the sink's DTCP Socket when the sink establishes a TCP connection to the source.

<service>://<host>:<port>/<path>/<FileName>.<FileExtention>?CONTENTPROTECTIONTYPE=DTCP1&DTCP1HOST=<host>&DTCP1PORT=<port>

Where:

CONTENTPROTECTIONTYPE is set to "DTCP1" where 1 represents a DTCP-IP version number that can be incremented in the future as needed.

DTCP1HOST specifies the IP address and **DTCP1PORT** specifies the port number of the DTCP Socket of the source device. The DTCP Socket for Remote Access is specified by **DTCP1RAPORT** which is the port number for RA-AKE along with the IP address specified with the **DTCP1HOST**.

V1SE 12.2.2 HTTP response /request

Content type of HTTP response / request[28] is set to DTCP application media type as follows:

Content-Type: application/x-dtcp1 ; DTCP1HOST=<host> ; DTCP1PORT=<port> ; CONTENTFORMAT=<mimetype>

[28] For example, HTTP POST request with "Expect: 100-continue" header.

V1SE 12.3 Header Field Definition for HTTP

The following header fields are defined for HTTP transfers.

V1SE 12.3.1 Range.dtcp.com

The Range.dtcp.com header is used in the same manner as the RANGE header defined in RFC 2616 except that range specification applies to the content before DTCP processing.

V1SE 12.3.2 Content-Range.dtcp.com

The Content-Range.dtcp.com header is used in the same manner as the CONTENT-RANGE header defined in RFC 2616 except that range specification applies to the content before DTCP processing.

V1SE 12.4 BLKMove.dtcp.com

The BLKMove.dtcp.com header is used to specify which K_{XM} is used in the Move Transmission process specified in V1SE 10.4.2. K_{XM}_label is a parameter of this header as follows:

BLKMove.dtcp.com:<K_{XM}_label>

<K_{XM}_label> is denoted in hexadecimal 2 digits.

V1SE 12.5 Alt-ExchangeKey.dtcp.com

The Alt-ExchangeKey.dtcp.com header is used to specify an Exchange Key to be used in a content transmission. This header is used to specify either the Session Exchange Key (K_S) or K_{XH0}[29], and not used for Exchange Key (K_X). In the content transmission using Exchange Key (K_X), this header is not used. The alt-exchange_key_label is a parameter of this header as follows:

Alt-ExchangeKey.dtcp.com:<alt-exchange_key_label>

<alt-exchange_key_label> is denoted in hexadecimal 2 digits. Note that the value of exchange_key_label for K_X is used for K_{XH0}. Source devices received this header with the value of exchange_key_label for K_X from a sink device may use K_{XH0} for content transmission to the sink device regardless of the value of the DOT field.

V1SE 12.6 CMI.dtcp.com

The CMI.dtcp.com header is used by sink devices to request content transmission using the CMI to source devices that support the CMI. The CC_req flag is a parameter of this header as follows:

CMI.dtcp.com:<CC_req>

<CC_req> is denoted in hexadecimal 1 digit and has the same semantics as the CC_req field in the section 8.3.4.10.

For non Copy-count content, <CC_req> is set to “0” by sink devices and ignored by source devices.

V1SE 12.7 RemoteAccess.dtcp.com

The RemoteAccess.dtcp.com header is used to specify K_R to be applied for the content transmission through the HTTP transfer including this header field. K_R label is a parameter of this header as follows:

RemoteAccess.dtcp.com:<K_R_label>

<K_R_label> includes the value of the exchange key label of K_R and is denoted in hexadecimal 2 digits.

[29] K_{XH0} is specified in the section B.3.1 of the Volume 1 Specification.

V1SE 12.8 Definition for UPnP AV CDS[30] Property

The following is defined for properties in UPnP AV CDS.

V1SE 12.8.1 DTCP.COM_FLAGS param

The DTCP.COM_FLAGS param is used in the 4th field of res@protocolInfo property to show static attribute of content regarding DTCP transmission. The DTCP.COM_FLAGS param is a 32 bit field, and the bit definition is as follows:

Bit 31: DTCP Movable
Bit 30: Move protocol specified in V1SE 10.4 is supported
Bit 29: Copy-count carried in CMI
Bit 28-0: Reserved (zero)

Bit 31 is set to one if associated content can be moved using DTCP. Bit 30 is also set to one if the content can be moved based on the Move protocol in V1SE 10.4. When only bit 31 is set to one, the Move protocol[31] in V1SE 10.4 cannot be used. Reserved bits are set to zero. Devices refer to the reserved bits ignore the value.

Bit 29 is set to one if the CC field has a value more than one when used in Move Transmission.

The 32 bits value of DTCP.COM_FLAGS param is denoted in hexadecimal 8 digits.

V1SE 12.8.2 res@dtcp:uploadInfo

The res@dtcp:uploadInfo property is used to show how the content is uploaded using DTCP. The res@dtcp:uploadInfo property is 32 bits field, and bit definition is as follows:

Bit 31: Content will be moved using DTCP Move
Bit 30: Move protocol specified in V1SE 10.4 will be used
Bit 29: Copy-count carried in CMI
Bit 28-0: Reserved (zero)

Bit 31 is set to one if associated content will be moved using DTCP. Bit 30 is also set to one if the move will be executed based on the Move protocol in V1SE 10.4. When only bit 31 is set to one, the Move protocol[31] in V1SE 10.4 is not used.. Reserved bits are set to zero. Devices refer to the reserved bits ignore the value.

Bit 29 is set to one if the CC field has a value more than one when used in Move Transmission.

The 32 bits value of res@dtcp:uploadInfo is denoted in hexadecimal 8 digits.

The definition of XML namespace whose prefix is "dtcp:" is "urn:schemas-dtcp-com:metadata-1-0/".

V1SE 12.8.3 res@dtcp:RSRegiSocket

The res@dtcp:RSRegiSocket property is used to describe one or more DTCP Sockets for the Remote Sink Registration. The res@dtcp:RSRegiSocket property is composed of a comma-separated list of the Sockets, and included in the first item in a CDS:Browse response. If all items in a CDS:Browse response are served by a single UPnP Media Server the res@dtcp:RSRegiSocket property has only one Socket, otherwise the Socket of each UPnP Media Server is listed in the res@dtcp:RSRegiSocket property. [e.g. <host1>:<port1>,<host2>:<port2>]

The definition of XML namespace whose prefix is "dtcp:" is "urn:schemas-dtcp-com:metadata-1-0/".

[30] Refer to UPnP ContentDirectory:2 document.

[31] Without using this Move protocol, move of content based on Exchange key (K_X) may be performed as specified in V1SE 4.27.