Document Number 2009011

HANA 2.1 *Content Services Draft*
TC-2009-0001
Version 1.0

November 10, 2009

Sponsored by:

1394 Trade Association

Accepted for publication by:

1394 Trade Association Board of Directors

Abstract:

Keywords:

Must, Must not, Required, Shall, Shall not, Should, Should not, Recommended, May, and Optional

1394 Trade Association Specification

1394 Trade Association Specifications are developed within Working Groups of the 1394 Trade Association, a non-profit industry association devoted to the promotion of and growth of the market for IEEE 1394-compliant products. Participants in Working Groups serve voluntarily and without compensation from the Trade Association. Most participants represent member organizations of the 1394 Trade Association. The specifications developed within the working groups represent a consensus of the expertise represented by the participants.

Use of a 1394 Trade Association Specification is wholly voluntary. The existence of a 1394 Trade Association Specification is not meant to imply that there are not other ways to produce, test, measure, purchase, market or provide other goods and services related to the scope of the 1394 Trade Association Specification. Furthermore, the viewpoint expressed at the time a specification is accepted and issued is subject to change brought about through developments in the state of the art and comments received from users of the specification. Users are cautioned to check to determine that they have the latest revision of any 1394 Trade Association Specification.

Comments for revision of 1394 Trade Association Specifications are welcome from any interested party, regardless of membership affiliation with the 1394 Trade Association. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally, questions may arise about the meaning of specifications in relationship to specific applications. When the need for interpretations is brought to the attention of the 1394 Trade Association, the Association will initiate action to prepare appropriate responses.

Comments on specifications and requests for interpretations should be addressed to:

Editor, 1394 Trade Association
315 Lincoln, Suite E
Mukilteo, WA 98275
USA

Published by

1394 Trade Association
315 Lincoln, Suite E
Mukilteo, WA 98275 USA

Printed in the United States of America

Document History

Version	Date	Editor(s)	Comments
.101	August 2008	Glenn Deen, IBM	Initial document Separated from HANA CP Common Book
.200	November 2008	Glenn Deen, IBM	Documented edited and update to reflect final changes in HANA Content Protection Specification
1.0	January 2009	Bill Rose, WJR Consulting	Add document #. Change revision to 1.0

Table of Contents

1 Introduction

1.1 Purpose and Scope

This document details services related to management of content information and access such as play lists. This is not a content protection document, that topic is covered in the companion HANA document *Content Protection Specification*.

1.2 Notice

Use of this specification and the associated cryptographic materials and patents needed to implement it require one or more licenses be obtained by adoptees.

1.3 Related Documents

This document is part of the HANA design specification which consists of this book, the HANA Design Guideline, and the HANA Content Protection Common Book.

1.4 References

The following standards contain provisions that, through reference in this text, constitute normative provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below.

[1] High-Definition Audio-Visual Network Alliance (HANA) Design Guideline 2.0

[2] HANA Content Protection Specification TC-2008-0001

1.5 Keywords

This document follows the convention set out in RFC 2119:

```
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
document are to be interpreted as described in RFC 2119.
```

Several keywords are used to differentiate levels of requirements as follows:

Shall or Must or Required

Indicate a mandatory requirement. Designers are required to implement all such mandatory requirements to assure interoperability with other products conforming to this standard. The addition of “Not” to either of these words indicates the prohibition of the specified requirement.

Should or Recommended

Denote flexibility of choice with a strongly preferred alternative, which may be ignored when the circumstances and implications of ignoring them are understood. The negation “Should not”, or “Not recommended” denote the flexibility of choice with a strongly preferred alternative which may be ignore under certain circumstances.

May or Optional

Indicate flexibility of choice with no implied preference.

1.6 Notation

1.6.1 Decimal Numbers

Decimal numbers are expressed as the value using the digits 0-9 (e.g. 170) without any additional notation.

1.6.2 Binary Numbers

Binary numbers are expressed using the digits 0 and 1 with the least significant digit on the right with a subscript of 2. e.g. 170 is expressed as 10101010_2

1.6.3 Hexadecimal Numbers

Hexadecimal numbers are expressed as the digits 0-9 and A-F with a subscript of 16. e.g. 170 is expressed as AA_{16}.

1.6.4 Bit Ordering

Unless explicitly noted otherwise values expressed as an array of bits are numbered such that for an array of n bits the least significant bit is numbered 0 and the most significant bit is numbered n-1.

1.6.5 Byte Ordering

Unless explicitly noted otherwise multi-byte values must be stored in big-endian form meaning that the first stored by holds the most significant byte of the value. i.e. 80FE16 is stored

1.6.6 Operators

The following notation will be used for bitwise and arithmetic operations:

$[x]_{msb\_z}$	The most significant z bits of x.	$[10010000_2]_{msb\_4} = 1001_2$
$[x]_{lsb\_z}$	The least significant z bits of x.	$[10010000_2]_{lsb\_4} = 0000_2$
$[x]_{y:z}$	The inclusive range of bits between bit y and bit z in x.	
$\sim x$	Bit-wise inversion of x.	
$x \parallel y$	Ordered concatenation of x and y.	
$x \oplus y$	Bit-wise Exclusive-OR (XOR) of two strings x and y.	
$x + y$	Modular addition of two strings x and y.	
$x \times y$	Multiplication of x and y.	
$x - y$	Subtraction of y from x.	

The following assignment and relational operators will be used:

=	Assignment
==	Equivalent (Equal to)
!=	Not equal to
<	Less than
>	Greater than

<= Less than or equal to

>= Greater than or equal to

1.7 Nomenclature

1.7.1 Abbreviations

AACS	Advanced Access Content System
AES	Advanced Encryption Standard (FIPS Publication 197)
AT	Authorization Table
DCL	Device Content List
MAC	Message Authentication Code
MKB	Management Key Block
NIST	National Institute of Standards and Technology

1.7.2 Definitions

Term	Abbrev	Description
Device ID		A 128 bit ID associated to a HANA device by the HANA Licensing Authority
Cluster ID		An 128 bit ID assigned to a HANA Content Cluster to distinguish it from other clusters.
Content Cluster		A collection of one or more authorized HANA devices which are bound together by a common AT and Cluster ID.
Content ID		A 128 bit identifier used to identify a content item.
Replica ID		A 128 but identifier used to distinguish between instances of a content item

2 Content Directory

2.1 Introduction

The Content Directory is a mechanism HANA devices make use of to make available to other HANA devices what content items they have. This can include both content physically stored in the device, and also content which reside on a storage medium outside of the device (see section 10 Bound Content of the HANA Content Protection Specification) for example on a network attached hard drive on which has content managed by the device

The Content Directory is not directly a playlist instead it is a means by which a HANA device MAY publish the list of content it holds, and by which a HANA device MAY acquire the list of content another HANA device holds. This can be used to construct playlists for individual devices, or even the entire cluster of devices. Construction, presentation, and advanced playlist function is out of the scope of the Content Directory architecture detailed here.

The Content Directory provides a standard representation of the list of content items on a device, a standard messaging definition for exchanging the list of content items, and an update mechanism for propagating changes in a device's Content Directory.

The Content Directory is not a secure content information database, and as such is NOT suitable for tracking of content licenses exported or deleted from the Content Cluster.

To avoid overloading device which have limited memory and processing ability, a device which supports Content Directory access to its content list is only REQUIRED to provide information on content which the device itself manages. Devices MAY provide content information about all content in a HANA network.

A device which implements supports Content Directory function maintains a list of content items, each identified by a Content ID. Access to this list by other devices on the network is done by device receiving a *getDCL* , get device content list, message. The device will respond to the message with the entire list of content it knows about. Also provides access to a compressed representation of its content list which is access by sending it a sync*DCL, synchronize device content list,* message. This message causes an exchange a compressed representation of the Device Content List in the form of a binary tree of hashes representing segments of the Device Content List held on each device. This permits devices to quickly check other devices for content items they share, and to efficiently check for changes in content lists.

One additional feature of the Content Directory is that a device keeps in its Device Content List a record of what other devices also have a copy of the content it has in its Device Content List. This permits devices to locate instances of content, and for playlist managers to quickly reflect content items which are no longer available when the device they are on is removed from the network.

2.2 Device Content List Data

The Device Content List consists of an entry for each content item consisting of a Content ID element, and a list of known replicas of the content item. The list of replicas includes all replicas held on the device, and all other devices in the HANA Content Cluster.

Each content item held on a device shall appear in the device's Device Content List

When an item is deleted from the local device, its entry shall be removed from the DCL.

The DCL and the aggregated DCL for the cluster only contain the content which is currently on the device, or in the cluster respectively. It is NOT used to track content for which the license has been exported. That is the role of the Content ID entry in the AT.

2.2.1 Item Entry

The DCL contains the following information for each content item:

Content ID

Replica List consisting of:

Device ID of the device holding the replica

Replica ID of each replica instance

Each copy instance of a content item is assigned a Replica ID which is unique in the set of Replica IDs for a particular Content ID. A consequence of this is that the content list on a device shall include all replicas residing on the device itself

2.2.1.1 Content ID

The Content ID is a 128 bit value which identified the content item. It is the AES 128 hash of the Content URN. The Content URN is held in the Protected Fixed Area Field of the Content Header as defined in Section 10.4.3.1 of the HANA Content Protection Specification.

Note: The Content ID in the Protected Fix Area Field is NOT the same as the Content ID used in the Device Content List.

2.2.1.2 Device ID

This is the 128 bit Device ID of the HANA device as defined in Section 6.3 of the HANA Content Protection Specification.

2.2.1.3 Replica ID

This is the 128 bit identifier used to distinguish different instances of a content item from one another. It is defined in Section 10.4.3.1 of the HANA Content Protection Specification.

2.2.2 Replica Instance List

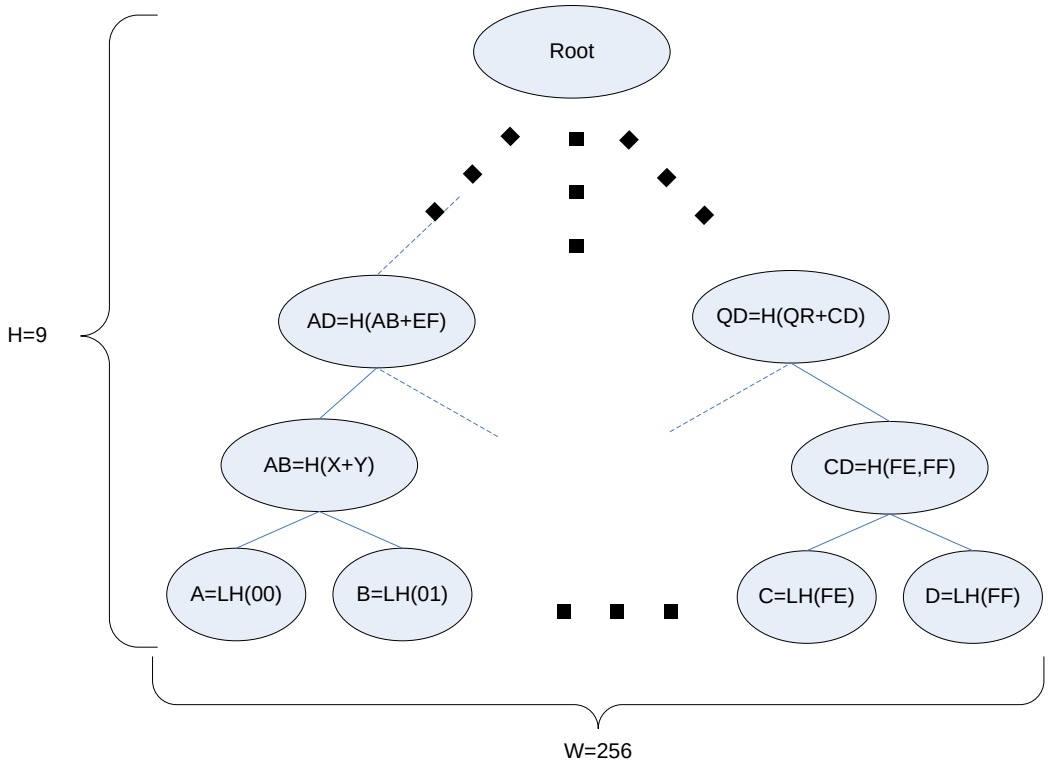
The replica instance list contains the list of all replicas of the content in the cluster known to the device

2.2.3 Device Content List Hash

Devices compute a compressed representation of the state of the Device Content List in the form of a tree of hashes of segments of the Device Content List. The tree is computed as follows:

The list of the Content IDs for content held on the device is segmented into buckets based on the first two bytes of the Content ID. A hash tree is then calculated with the hash of list Content IDs in each bucket forming the leaf nodes of the tree. Each higher node is the AES_H hash of the concatenation of the two lower nodes. This is repeated upwards to the root node.

The hash tree has 256 leaf nodes representing the lists of Content IDs group by the first two characters, and has a height of 9 levels for a total of 511 nodes. Each node is 128 bits (16 bytes) in size, making the entire tree 8176 bytes.



2.3 Device Content List Protocol Messages

The schema for the Content Directory messages is included in the supplemental material at the end of this document.

2.3.1 getDCL Message

The *getDCL* message is send to a device to retrieve the Device Content List held on the device. It supports requesting either one or more subsets of the list, or the entire Device Content List for the device.

2.3.1.1 Message Elements:

Name	Required	Meaning
clusterid	yes	128 bit id of the Content Cluster for which the Device Content List is being requested. If the Cluster ID of the receiving device does not match this, the receiving device SHALL return a "rejected:refused" response.
List	no	If present it will contain a list of the buckets of the hash tree to return the content list of. The list SHALL be white space separated If not present the entire content list held by the device shall be returned

2.3.1.2 Rejection Codes

Response	Meaning
rejected: invalid	The message had missing or invalid elements, or the device does not support getDCL requests
rejected: refused	The receiving device is a state where it is not accepting messages or it has refused the request OR The Cluster ID in the request did not match the Cluster ID of the receiving device.

2.3.1.3 Example

Sent

```
<message>
  <getDCL>
  <clusterid>c23854c25c9b4bff9f29c00cb7c0290b</clusterid>
    <list>03 af</list>
  </getDCL>
</message>
```

Successful response

```
<response>
  <getDCL>
    <content id="032212aabbccddee032212aabbccddee">
      <replica id="123412345678abc9def05678abc9def0"
               device="1122334455667788aabbccddeeff00"/>
    </content>
    <content id="03123412341234aabbcceeddffeeaabb">
      <replica id="12345678abcdef012345678abcdef0f0"
               device="1122334455667788aabbccddeeff00"/>
      <replica id="32345678abcdef112345678abcdef0af"
               device="1122334455667788aabbccddee1234"/>
    </content>
    <content id="af11223344556677889900aabbccddee">
      <replica id="12345678abcdef0012345678abcdef00"
               device="1122334455667788aabbccddeeff00"/>
      <replica id="32345678abcdef112345678abcdef0af"
               device="1122334455667788aabbccddee5678"/>

    </content>
   </getDCL>
  </response
```

Failure response

```
  <response>
    <rejected reason=rejection-code/>
  </response>
```


2.3.2 syncDCL Message

The sync*DCL* message is send to a device to retrieve the hash for the Device Content List held on the device.

Message Elements:

Name	Required	Meaning
clusterid	yes	128 bit id of the Content Cluster for which the Hash of the Device Content List is being requested. If the Cluster ID of the receiving device does not match this, the receiving device SHALL return a "rejected:refused" reponse.

2.3.2.1 Rejection Codes

Response	Meaning
rejected: invalid	The message had missing or invalid elements, or the device does not support syncDCL requests
rejected: refused	The receiving device is a state where it is not accepting messages or it has refused the request OR The Cluster ID in the request did not match the Cluster ID of the receiving device.

2.3.2.2 Example

Sent

```
<message>
  <syncDCL>
       <clusterid>c23854c25c9b4bff9f29c00cb7c0290b</clusterid>
       <hash>encoded hash tree</hash>
  </syncDCL>
</message>
```

Successful response

```
<response>
  <syncDCL>
    <hash>encoded hash tree</hash>
  </syncDCL>
</response>
```

Failure response

```
<response>
  <rejected reason="rejection-code"/>
</response>
```

Rejection Codes

Response	Meaning

rejected: invalid	The message had missing or invalid elements, or the device does not support syncDCL requests
rejected: refused	The receiving device is a state where it is not accepting messages or it has refused the request OR The Cluster ID in the request did not match the Cluster ID of the receiving device.

2.3.3 getContentURN Message

The *getContentURN* message is send to a device to retrieve the Content URN for a Content ID entry in the Device Content List held on the device. It supports requesting either one or more mapping at a time. The Content URN can be used by a device to obtain extended information about a content item from an content information source such as an ISAN repository for works.

The Content URN returned is defined in Section 12 of the HANA Content Protection Specification.

2.3.3.1 Message Elements:

Name	Required	Meaning
clusterid	yes	128 bit id of the Content Cluster for which the Content item URN Mapping is being requested. If the Cluster ID of the receiving device does not match this, the receiving device SHALL return a "rejected:refused" response.
getURN	no	Multiplicity: appears 0 or more times If present it will contain the Content ID to return the Content URN's for. If not present the Content URNs for all content items held by the device shall be returned Attribute: id – the Content ID of the item

2.3.3.2 Rejection Codes

Response	Meaning
rejected: invalid	The message had missing or invalid elements, or the device does not support getContentURN requests
rejected: refused	The receiving device is a state where it is not accepting messages or it has refused the request OR The Cluster ID in the request did not match the Cluster ID of the receiving device.

2.3.3.3 Example

Sent

```
<message>
   <getContentURN>
      <clusterid>c23854c25c9b4bff9f29c00cb7c0290b</clusterid>
     <getURN id="032212aabbccddee032212aabbccddee</getURN>
   </getContentURN>
</message>
```

Successful response

```
<response>
   <getContentURN>
     <content id="032212aabbccddee032212aabbccddee"
        urn="urn:isan:0000-0000-9E59-0000-O-0000-0000-
    :urn:ASCCT:090912780073120341298871O"
     </content>
    </getContentURN>
   </response
```

Failure response

```
   <response>
     <rejected reason=rejection-code/>
   </response>
```

2.4 Device Content List Management Events

When a Device Content List changes the device contacts the other devices on the network and sends a *syncDCL* message. A device MAY choose to delay sending the *syncDCL* message if the device is performing a number of changes such as a user importing content and/or deleting content items. The device would instead send a single *syncDCL* reflecting the final state of the Device Content List.

2.4.1 Deletion of Content

When an item is deleted on a device, its replica instance entry shall be removed from its Device Content List.

When an item has no more entries in its replica instance list, then the entry for the content ID may be deleted from the list.

2.4.2 Addition of Content

When an item of content is added, its SHALL have its Content ID and the local replica instance add to the Device Content List.

2.5 Use Cases

The following use cases illustrate interactions between devices.

2.5.1 Change Propagation

A device notifies the other devices on the network when changes are made to its Device Content List. Such notification MAY hold an aggregated set of changes.

The device regenerates its content list hash tree and sends it to the other devices on the network via the *syncDCL* message.

Each device then compares the hash with the last previously received hash from the other device. It identifies which leaf nodes have changes since the last synchronization, and if the changed leaves nodes are also leaf nodes representing content that the device also holds or is tracking the device will then obtain the Device Content List from the device using the *getDCL* message to update its own Device Content List.

Finally, the device replaces its stored copy of the hash tree sent by the changed device with the one it received from the *syncDCL* message.

2.5.2 Device List Acquisition

A device can obtain the list of content items held on another device with the *getDCL* message. This query can be to obtain either the entire list, a subset of the list

The data returned is the information about the item held in the device's content list.

2.5.3 Cluster Wide Playlist Composition

A device can build a list of all the available content on the HANA network by sending a *getDCL* message to each HANA device and aggregating the list.

2.5.4 Synchronization on Network Connect/Reconnect

A device SHALL attempt to synchronize its Device Content List when it connects/reconnects to the HANA network. It SHALL send *syncDCL* messages to the HANA devices on the network to start the synchronization process with each device.

Supplemental Information

3 Message Schema

The following are the message schema use for the messages exchanged by HANA devices for performing Cluster Management and Content Directory transactions.

3.1 Device Content List Message Schema

NOTE : This messsage schema is not currently up to date – TODO – upon agreement of main document body – update the schema to reflect the final decision of the HANA TWG -

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.hanaalliance.org/hana/21/dcl"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:ts="http://www.hanalliance.org/hana/21/types"
   xmlns:ds="http://www.hanaalliance.org/hana/21/dcl">
   <xsd:import namespace="http://www.hanalliance.org/hana/21/types"
      schemaLocation="hanatypes.xsd">
   </xsd:import>

   <xsd:element name="message" type="ds:dclMessageType"></xsd:element>
   <xsd:element name="response"
type="ds:dclMessageResponseType"></xsd:element>

        <xsd:complexType name="dclMessageType">
                <xsd:choice minOccurs="1">
                        <xsd:element name="getDCLHash"
                                        type="ds:getDCLHashMessageType"/>
                        <xsd:element name="getDCL"
                                        type="ds:getDCLMessageType"/>
                </xsd:choice>
        </xsd:complexType>

   <xsd:complexType name="dclMessageResponseType">
      <xsd:choice minOccurs="1">
         <xsd:sequence>
            <xsd:element name="hash"
               type="ds:getDCLHashResponseMsgType">
            </xsd:element>
         </xsd:sequence>
         <xsd:element name="dcl" type="ds:getDCLResponseType">
         </xsd:element>
      </xsd:choice>
   </xsd:complexType>


   <xsd:complexType name="getDCLHashMessageType">
      <xsd:element name="clusterid" type="ts:cluseridType"></xsd:element>
   </xsd:complexType>

   <xsd:complexType name="getDCLMessageType">
      <xsd:element name="clusterid" type="ts:cluseridType"></xsd:element>
      <xsd:element name="list" minOccurs="0">
         <xsd:simpleType>
            <xsd:list>
               <xsd:simpleType>
```



```
            <xsd:restriction base="xsd:hexBinary">
              <xsd:length value="2"></xsd:length>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:list>
      </xsd:simpleType>
      <xsd:sequence>
        <xsd:element name="i" minOccurs="0"
          maxOccurs="unbounded" type="ts:contentidType">
      </xsd:sequence>
    </xsd:element>
</xsd:complexType>

<xsd:complexType name="getDCLHashMessageResponseType">
  <xsd:choice>
    <xsd:element name="hash" type="xsd:hexBinary">
      <xsd:annotation>the MKB</xsd:annotation>
    </xsd:element>
    <xsd:element name="rejected" type="cs:rejectReasonType">
    </xsd:element>
  </xsd:choice>
</xsd:complexType>

<xsd:complexType name="getDCLMessageResponseType">
  <xsd:choice>
    <xsd:element name="DCL">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="content">
            <xsd:complexType>
              <xsd:attribute name="id"
                type="ts:contentIDType">
              </xsd:attribute>
              <xsd:sequence>
                <xsd:element name="replica"
                  minOccurs="0" maxOccurs="unbounded">
                  <xsd:complexType>
                    <xsd:attribute name="id"
                      type="ts:contentIDType">
                    </xsd:attribute>
                  </xsd:complexType>
                </xsd:element>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:annotation>
        the requested DCL entries
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="rejected" type="cs:rejectReasonType">
    </xsd:element>
  </xsd:choice>
</xsd:complexType>
```



```
</schema>
```

