© 2009 Universal Music Group. All rights reserved. No part of this specification may be reproduced or utilized in any form or by any means, including electronic means, without express permission from Universal Music Group. This specification is made available under your existing download agreement with the Universal Music Group. For additional information, contact Universal Music Group at: uits-legal@umusic.com.

1. Introduction

The Unique Identifier Technology Solution (UITS) is an industry initiative that describes a common way to embed metadata into unprotected media files so that it is possible to detect when the metadata is modified. The UITS payload is primarily designed for audit purposes by record labels, artists, and others (Content Owners), but is flexible enough for other uses. It is understood that the embedded metadata can be removed from files, though there should not be any reason to do so.

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.

2. Technical Goals

From an audit perspective, the UITS technical goals are three-fold:

1) Distinguish between different legitimately acquired instances of a single track based on the sale. This is accomplished using an identifier. This means that if user #1 buys a track and user #2 buys the same track, each track instance has a different identifier. At the same time, if for whatever reason a user is able to download a single track multiple times from a single sale (say, due to a pre-specified business arrangement), those downloads should share the same identifier. If, on the other hand, a user buys a given track multiple times on separate occasions, then each purchase must be differentiated either by identifier or by a combination of identifier and timestamp.
2) Identify legitimate payloads and detect when a payload has been modified. It is not necessary to determine what the original payload was, simply that the original payload has been altered.
3) Identify the original distributor, date of sale, and track solely from the payload so that the payload can be verified.

3. Tamper Detection

Tamper detection is accomplished by cryptographically signing a collection of metadata, then inserting both the metadata and signature into the file. Together, the metadata and signature are called the UITS payload. To enable verification that the payload is embedded into the correct file, a cryptographic (one-way) hash or fingerprint is created of the encoded media (e.g. the audio or video portion of the file) and included in the

payload. For example, this means the descriptive metadata in an MP3 can be modified but the UITS payload can still be matched with the track. Furthermore, the hash or fingerprint can be pre-computed rather than generated at the time of sale. If any part of the payload is modified, this will become apparent when the signature is checked.

4. UITS Identifier Options

This section describes the different UITS identifier options open to distributors. One or both of these methods MUST be used. Unless other arrangements have been made, these transaction and/or User IDs MUST be reported back to the Content Owner in a standard marketing sales report.

4.1. OPTION 1: Embedded User IDs

In this implementation, the ID component in the UITS payload consists of user-specific information, optionally encoded such that it cannot be directly traced back to the user. The value MUST be static (for that distributor), meaning if a given user buys multiple tracks, the same information is embedded into each. Examples of user-specific information might include: a) an email address or obfuscated email address, b) obfuscated login, or c) a user's service ID number or obfuscated ID number. Any obfuscation method used must be deterministic. If a user buys a single track multiple times, then different purchase timestamps MUST be used. If a user buys a gift for someone else, then the user ID should be that of the recipient rather than the buyer. If no such recipient ID exists (because the recipient is anonymous), then a random, unique ID should be created.

4.2. OPTION 2: Embedded Unique Transaction IDs

In this implementation, the ID component of the UITS payload is a globally unique identifier (for that distributor) that identifies a specific customer transaction. This means that if the user purchased 3 tracks and 2 albums, all of the individual files would contain the same transaction ID. If a user is able to download a purchase multiple times, each download MUST contain the same transaction ID. If the user buys the same track multiple times, each purchase MUST have different transaction IDs.

5. Technical Specifications

In all cases, the UITS payload has two primary components: metadata and the associated signature. The syntax used for organization is XML, and there is no special requirement that element or attribute names be capitalized. At a high level, the payload structure is

```
<UITS>
    <metadata></metadata>
    <signature></signature>
</UITS>
```

For all of the element descriptions below, the layout is:

Title

REQUIRED/OPTIONAL. Description of the element and a list of any attributes and valid attribute values.

`<Example>`

5.1. Encoding

Data contained within a UITS document is encoded using either UTF-8 or 16, as noted by the XML processing header. Note that the only legal white space characters that can appear within the <UITS> element are spaces, tabs, carriage returns, and line feeds.

```
<?xml version="1.0" encoding="UTF-8"?>
```

5.2. UITS

REQUIRED. The UITS element contains a <metadata> element and a <signature> element. The version of the UITS specification is denoted by the namespace. The current version is 1.1. Within the UITS element, namespace notations MAY be included.

```
<UITS xmlns:uits="http://www.udirector.net/schemas/2009/uits/1.1">
```

5.3. Metadata

REQUIRED. This element wraps all of the support metadata below.

```
<metadata>
```

The metadata element contains the following sub-elements, which should be included in the UITS payload in the order listed below.

5.3.1. Nonce

REQUIRED. The nonce is a random value generated at the time of sale whose Base64 encoded length is 8 digits. The nonce is used to randomize the hash computation of the signature value for security purposes.

```
<nonce>QgYnkgYS</nonce>
```

5.3.2. Distributor ID

REQUIRED. This is the retailer or distributor's name or an associated globally unique identifier in clear text. Note that this does not identify the CDN or entity generating and/or embedding the UITS payload, but rather the name of the company that has the record of sale (or its agent) and issues sales reports back to the Content Owner. A distributor that operates in multiple territories may use different IDs in each territory.

```
<Distributor>Music Retailer</Distributor>
```

5.3.3. Date of transaction

REQUIRED. The date and time of the purchase (not download or signature) in ISO 8601 format. For example, a file purchased at 14:15 on the 30th August 2008 in the UK should include the string:

```
<Time>2008-08-30T13:15:04Z</Time>
```

Either UTC, or local time plus offset MUST be used. Further documentation on ISO 8601 can be found at http://www.w3.org/TR/NOTE-datetime

5.3.4. Product ID

REQUIRED. The Product ID identifies the parent collection from which the current item originates. If the entire collection has been purchased or completed during the transaction, then this is indicated through the "completed" attribute. The ProductID type is passed as an attribute. Valid ProductID Types currently include "UPC" or "GRID".

For example, if a user purchases a single track from an album, the Product ID identifies the associated album and the completed attribute is set to "false". If instead the user purchases the entire album, the Product ID still identifies the associated album, but the completed attribute is set to "true." Some retailers support a "complete my album" sale, and for those transactions the completed attribute would also be set to true. Only when the entire collection is completed in the same transaction is the completed attribute set to true.

```
<ProductID type="UPC"
completed="false">00602517758056</ProductID>
```

5.3.5. Asset ID

REQUIRED. The Asset ID identifies an individual element of a product, such as a track or video. The AssetID type is passed as an attribute. Valid AssetID types currently include "ISRC". Delimiters such as dashes MUST NOT be included.

```
<AssetID type="ISRC">ES1700800499</AssetID>
```

5.3.6. Transaction ID

Note: at least one of Transaction ID or User ID is REQUIRED. The transaction ID is the unique identifier for the transaction. Currently the only valid version is "1".

Obfuscation MAY be used so long as the algorithm is deterministic, meaning the same output is calculated given the same input. For example, a standard cryptographic HMAC algorithm may be used. A list MUST be maintained of all issued identifiers to prevent collisions and to assist with breach detection.

```
<TID version="1">39220345237</TID>
```

5.3.7. User ID

Note: at least one of Transaction ID or User ID is REQUIRED. The User ID is the unique identifier for the user. Currently the only valid version is "1".

Obfuscation MAY be used so long as the algorithm is deterministic, meaning the same output is calculated given the same input. For example, a standard cryptographic HMAC algorithm may be used. A list MUST be maintained of all issued identifiers to prevent collisions and to assist with breach detection.

```
<UID version="1">A74GHY8976547B</TID>
```

5.3.8. Media Identifier

REQUIRED. A hash of the media (e.g. the audio, video, etc.) portion of the file MUST be included, such that the UITS payload can be directly tied to the file associated with it. The hash type is passed as an attribute, and the currently valid type is "SHA256". Metadata fields MUST NOT be included in the hash computation, because it must be possible for users to update standard ID3 or similar tags without affecting the media hash. Note that the hash can be pre-computed so that it is not necessary to compute it at the time of sale.

Instructions for MP3s: The hash value for the audio part of the MP3 is calculated using the hash algorithm over all of the audio frames within the file in the order in which they appear in the file. The audio frames all start with a 12-bit syncword, their frame size is calculated in the standard manner from the bitrate, sampling and padding. Frames that are not audio (such as frames containing ID3 tags) are excluded. In addition, one type of audio-related frame must be identified and not included in the cryptographic hash, specifically, older Xing variable byte rate data frames. They are documented at: http://gabriel.mp3-tech.org/mp3infotag.html and http://www.codeproject.com/KB/audio-video/mpegaudioinfo.aspx.

Note that future versions of UITS may support fingerprints, embedded watermarks, or other techniques for identifying audio.

```
<Media algorithm="SHA256">A675BD878C8658C99A…</Media>
```

5.3.9. URL

OPTIONAL. The URL field is used for links that are worth transporting in a cryptographically signed manner. The type field specifies the URL type and currently uses a subset of ID3v2.3 tag names. Options for the type field include WCOM, WCOP, WOAF, WOAR, WOAS, WORS, WPAY, and WPUB, which have the meaning described in the ID3 specification. In addition, a legal URL type is KeyURI, which identifies the public key needed to validate the signature (see the Implementation Details section below). More than one URL element MAY be included.

```
<URL type="WPUB">http://www.universalmusic.com/</URL>
```

5.3.10. Parental Advisory

OPTIONAL. The Parental Advisory field is used to indicate the parental advisory status for the track. The three valid values are "unspecified", "explicit", and "edited". The absence of the PA element can be interpreted as "unspecified".

- **unspecified** means no information about the PA status is being communicated. Absence of a PA field is the same as unspecified.
- **explicit** means that the content to which it is applied would, if sold as a physical product or in its originally released compilation, warrant the application of a parental advisory mark or sticker in the region for which it is originally intended.
- **edited** means that the content is an edited version of original content to which a parental advisory mark or sticker has been applied.

```
<PA>explicit</PA>
```

5.3.11. Copyright Status

OPTIONAL. This field can be used to communicate copyright information about the associated track. The value field can be one of these pre-defined values: "**unspecified**", "**allrightsreserved** ", "**prerelease** ", or "**other**".

- **unspecified** means no information about the copyright status is being communicated. Absence of a copyright field is the same as unspecified.
- **allrightsreserved** means all rights reserved unless otherwise expressly authorized by the content owner.
- **prerelease** means the same as all rights reserved, but additionally, that the content is pre-release material and not intended for release to the public.
- **other** means that the copyright element contains a URL that has more information about the copyright. For example, this might be used for a creative commons license.

```
<Copyright value="allrightsreserved"></Copyright>  or
<Copyright value="other"> http://tinyurl.com/awmp4f</Copyright>
```

5.3.12. Extra

OPTIONAL. The Extra field is used for miscellaneous metadata that is worth transporting in a cryptographically signed manner. The type field specifies the type of metadata and the value field specifies the value. More than one Extra element MAY be included.

```
<Extra type="Foo">Bar</Extra>
```

5.4. Signature

A cryptographic signature is used to verify the integrity of the <metadata> element. The contents of the <metadata> element are signed including the opening and closing <metadata> tags, and the resulting signature is placed in a <signature> element. The <signature> element is then appended after the end of the metadata element. The whole thing is then wrapped in a UITS element (see below for an example).

The signature block consists of five parts. The wrapping <signature> element, three attributes, and content representing the Base64 encoded result of the signature algorithm. The following attributes relate to the mechanism used for creating and interpreting the <signature> value.

- algorithm – identifies the signature and hash algorithms plus any critical configuration details such as padding used to generate the signature. Current options are "RSA2048" and "DSA2048" and distributors can pick either one.
    - RSA2048 means generating the signature with the RSA algorithm using RSASSA-PKCS1 –v1_5 padding as defined in RFC 3447, with a 2048 bit key. The SHA256 algorithm must be used to hash the metadata element.
    - DSA2048 means using the DSA signature algorithm as described in FIPS 186-3 with a 2048 bit key. The SHA224 algorithm must be used to hash the metadata element.
- canonicalization – identifies the XML canonicalization method used to normalize the data contained within the <metadata> element. Currently, the only method supported is "none". This means that when the hash is computed on the <metadata> element, the <metadata> element must be hashed exactly as it exists in the file including any white space characters (tabs, carriage returns, line feeds, and spaces) that it contains.
- keyID – an identifier associated with the public/private key pair used by the distributor or CDN for computing the signature value. The keyID is the SHA1 hash of the public key needed to validate the signature.
    - For RSA, the KeyID is the hash of the DER-encoded RSAPublicKey (as defined in RFC 3447).
    - For DSA, the KeyID is the hash of the DER-encoded DSAPublicKey (as defined in RFC 3279).

Here is an example signature, formatted for readability.

```
<signature algorithm="RSA2048"
     canonicalization="none"
     keyID="b2568ca44decce80066ece19bf47488b42d337ec">
UjBsR09EbGhjZ0dTQUxNQUFBUUNBRU1tQ1p0dU1GUXhEUzhi…
</signature>
```

6. Implementation Details

6.1. Keys

Partners are responsible for generating and maintaining the security of their public/private key pair. Private keys MUST be protected with the same care as highly confidential information. If a partner believes that a key may be compromised, they MUST immediately generate a new key pair and alert any affected content owners.

6.2. Payload creation

For security reasons, the payload MUST only be generated on the distributor's or CDN's servers.

The signature is generated by executing the hash function on the <metadata> block, signing the result using the private key associated with KeyId and encoding the result in Base64. The signature extends over the entire <metadata> element.

D1 = Canonicalize("<metadata>…</metadata>")
D2 = Hash(D1)
D3 = Sign(private_key, D2)
D4 = Base64encode(D3)

(Note: there is no canonicalization algorithm in the present version of the specification, so D1 results in no change to the contents of the <metadata> element).

The resulting signature is encoded in a <signature> block and combined with the original <metadata> block to create the full <UITS> element. The resulting UITS element MUST parse using the associated XML Schema.

For example, to perform steps D2-D4 above on a UITS <metadata> element in the file payload.txt, the following openSSL commands could be used (assuming the private key is in the file privateRSA.pem file):

```
openssl dgst -sha256 -out sig.bin -sign privateRSA.pem payload.txt
openssl base64 -in sig.bin -out signature.txt
```

The contents of the signature.txt file would then be inserted between the <signature></signature> tags.

6.3. Payload embedding process

If no download manager is used, the payload MUST be embedded on the distributor's or CDN's servers. If a download manager is used, then the payload MAY be embedded by client software, so long as no file is made available to end-users without a payload (i.e., the embedding must happen during the download process, prior to the user being able to access the file). Systems that support client-side embedding MUST be designed to prevent and detect unauthorized access or interference with the embedding process.

The specific area of the file where the UITS payload MUST be embedded varies by file type.

- MP3 - the payload MUST be embedded in a PRIV ID3 tag.

7. Example Payload

Note the hash and signature values are not real.

```
<?xml version="1.0" encoding="UTF-8"?>
<uits:UITS xmlns:uits="http://www.udirector.net/schemas/2009/uits/1.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <metadata>
      <nonce>QgYnkgYS</nonce>
      <Distributor>Music Retailer</Distributor>
```

```
      <Time>2001-12-17T09:30:47.0Z</Time>
      <ProductId type="UPC"
                 completed="false">00602517178656</ProductId>
      <AssetID type="ISRC">USUV70603512</AssetID>
      <TID version="1">39220345237</TID>
      <UID version="1">A74GHY8976547B</UID>
      <URL type="WPUB">http://www.umusic.com</URL>
      <Media algorithm="SHA256">
      d5b17cc1975d3095c6353f3fdced45ae867c06e02c1efb7c09662cdc796724b0
      </Media>
   </metadata>
   <signature algorithm="RSA2048"
         canonicalization="none"
         keyID="b2568ca44decce80066ece19bf47488b42d337ec">
iQEcBAEBAgAGBQJJ0S2XAAoJECY0BcYmEHZbpjcIAJNKgwFg93vvCxfn6cDVthZr
ERjp1gq1Nqt6vcBBP2zfdhgsA6eay4EMpM7K9o1cpN7UH1ksnFTpeniUrdfZs9FW
RTusFwTSHv82D1ZJQ3mvq0r25KKrtAB7FUz2G3XFP8zE/TdUCLjO7A0DhME9sytl
yZrPyZlraJGpfzUK9lRLY4IfupBnPDhav9Quycfoq2Kfnv2in/8PMvZdQH0Kim9b
z02gWsRijoMM7Vy7esBPLeXA++oOo28H7cnd9+BaZ5wtxbxXGg9G5pAWwdW45Ddi
2q+GpiUCqeHbqnucsHVfa2wyP7b6ASTUSzZHJEGle7UkHm057fCCa72n6vqTNfA=
=VYk6
   </signature>
</uits:UITS>
```


8. Verification

The <metadata> element MUST validate if it is directly copied and pasted into a file, treating white space as indicated in the canonicalization rules, and then checked against the signature hash.

To validate the <metadata> block:

D1 = Canonicalize("<metadata>…</metadata>")
D2 = Base64decode(signature)
D3 = Verify(public_key, D1, D2)

(Note: there is no canonicalization algorithm in the present version of the specification, so D1 results in no change to the contents of the <metadata> element).

For example, to do steps D2-D3 above and verify the signature using openSSL, you could do the following (assuming the payload was in payload.txt, the signature was in signature.txt, and the public key was in the pubRSA.pem file):

```
openssl base64 -d -in signature.txt -out sig.bin
openssl dgst –sha256 -verify pubRSA.pem -signature sig.bin payload.txt
```

Appendix – XML Schema Documentation

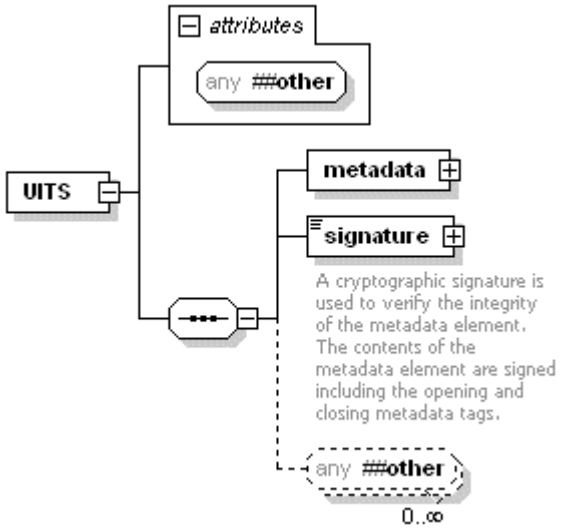
Schema **uits.xsd**

schema location: Z:\Projects\uits\lib\uits\uits.xsd
attribute form default:
element form default:
targetNamespace: **http://www.udirector.net/schemas/2009/uits/1.1**

Elements	Complex types
UITS	copyrightType
	metadataType
	signatureType

element **UITS**

diagram



namespace	http://www.udirector.net/schemas/2009/uits/1.1
properties	content complex
children	metadata signature
attributes	Name Type Use Default Fixed annotation

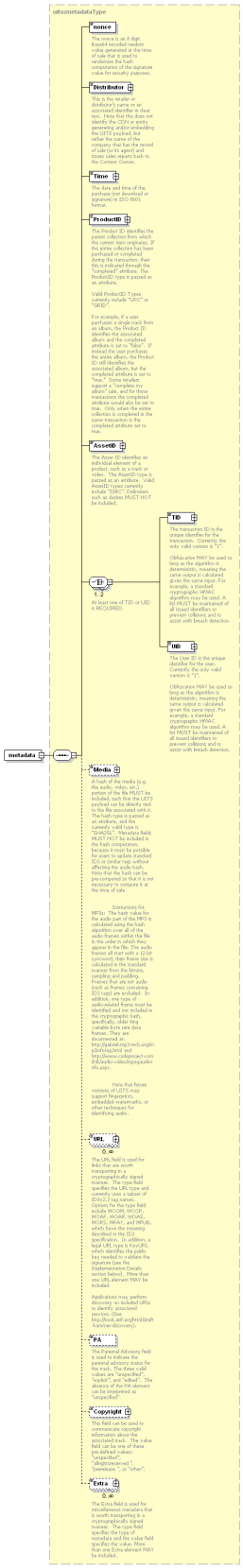
source

```
<xs:element name="UITS">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="metadata" type="uits:metadataType"/>
      <xs:element name="signature" type="uits:signatureType">
        <xs:annotation>
          <xs:documentation>A cryptographic signature is used to verify the integrity of
the metadata element.  The contents of the metadata element are signed including
the opening and closing metadata tags.</xs:documentation>
        </xs:annotation>
```

	`</xs:element>`
	`<xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>`
	`</xs:sequence>`
	`<xs:anyAttribute namespace="##other" processContents="lax"/>`
	`</xs:complexType>`
	`</xs:element>`

element **UITS/metadata**

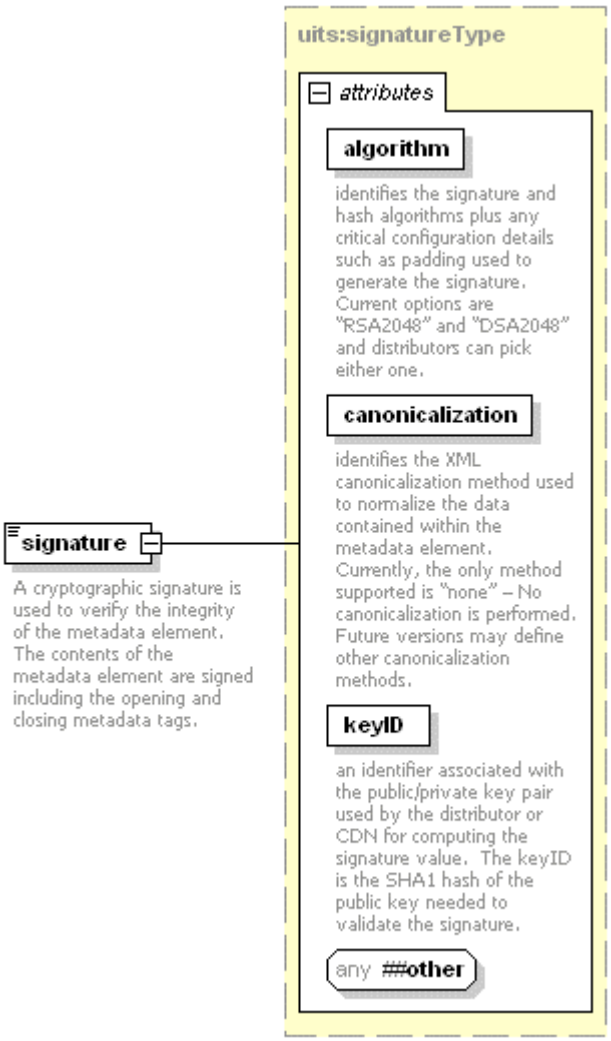
diagram



type	**uits:metadataType**
properties	isRef 0 content complex
children	**nonce** **Distributor** **Time** **ProductID** **AssetID** **TID** **UID** **Media** **URL** **PA** **Copyright** **Extra**
source	<xs:element name="metadata" type="uits:metadataType"/>

element **UITS/signature**

diagram



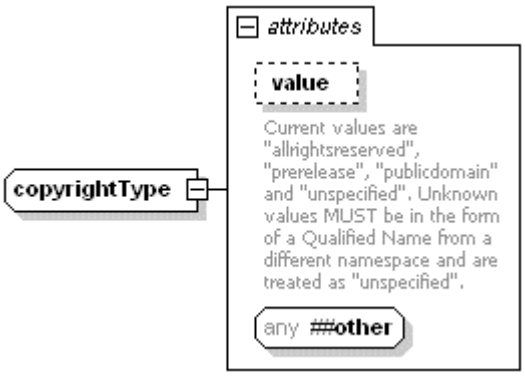
type	**uits:signatureType**
properties	isRef 0 content complex

attributes

Name	Type	Use	Default	Fixed	annotation
algorithm	**derived by: xs:NCName**	required			documentation identifies the signature and hash algorithms plus any critical configuration details such as padding used to

				generate the signature. Current options are “RSA2048” and “DSA2048” and distributors can pick either one.
	canonicalization	**derived by: xs:NCName**	required	documentation identifies the XML canonicalization method used to normalize the data contained within the metadata element. Currently, the only method supported is “none” – No canonicalization is performed. Future versions may define other canonicalization methods.
	keyID	**xs:NMTOKEN**	required	documentation an identifier associated with the public/private key pair used by the distributor or CDN for computing the signature value. The keyID is the SHA1 hash of the public key needed to validate the signature.
annotation	documentation A cryptographic signature is used to verify the integrity of the metadata element. The contents of the metadata element are signed including the opening and closing metadata tags.			
source	<xs:element name="signature" type="uits:signatureType"> <xs:annotation> <xs:documentation>A cryptographic signature is used to verify the integrity of the metadata element. The contents of the metadata element are signed including the opening and closing metadata tags.</xs:documentation> </xs:annotation> </xs:element>			

complexType **copyrightType**

diagram	
namespace	http://www.udirector.net/schemas/2009/uits/1.1
type	extension of **xs:anyURI**
properties	base xs:anyURI mixed true
used by	element **metadataType/Copyright**

attributes

Name	Type	Use	Default	Fixed	annotation
value	**derived by: xs:NCName**				documentation Current values are "allrightsreserved", "prerelease", "publicdomain" and "unspecified". Unknown values MUST be in the form of a Qualified Name from a different namespace and are treated as "unspecified".

source

```
<xs:complexType name="copyrightType" mixed="true">
 <xs:simpleContent>
  <xs:extension base="xs:anyURI">
   <xs:attribute name="value">
    <xs:annotation>
     <xs:documentation>Current values are "allrightsreserved", "prerelease",
"publicdomain" and "unspecified". Unknown values MUST be in the form of a
Qualified Name from a different namespace and are treated as
"unspecified".</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
     <xs:union>
      <xs:simpleType>
       <xs:restriction base="xs:NCName">
        <xs:enumeration value="unspecified"/>
        <xs:enumeration value="allrightsreserved"/>
        <xs:enumeration value="prerelease"/>
        <xs:enumeration value="other"/>
       </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
       <xs:restriction base="xs:QName"/>
      </xs:simpleType>
```

```
        </xs:union>
      </xs:simpleType>
    </xs:attribute>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:extension>
 </xs:simpleContent>
</xs:complexType>
```

attribute **copyrightType/@value**

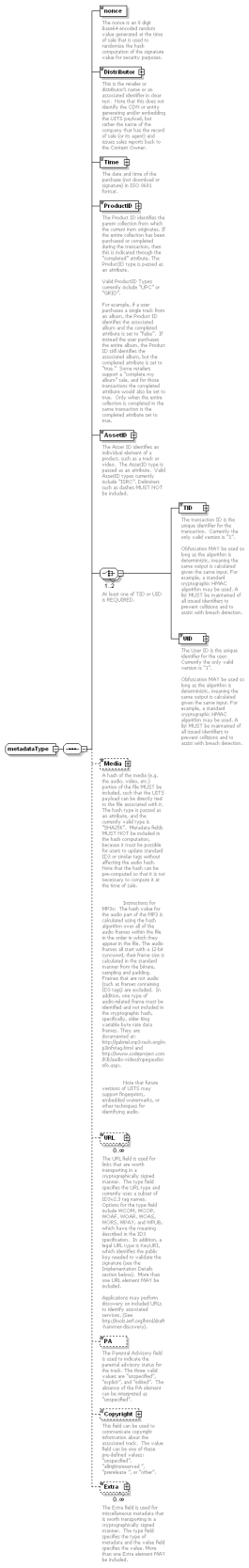
type	union of (restriction of **xs:NCName**, **xs:QName**)
properties	isRef 0
annotation	documentation Current values are "allrightsreserved", "prerelease", "publicdomain" and "unspecified". Unknown values MUST be in the form of a Qualified Name from a different namespace and are treated as "unspecified".

source

```
<xs:attribute name="value">
 <xs:annotation>
  <xs:documentation>Current values are "allrightsreserved", "prerelease",
"publicdomain" and "unspecified". Unknown values MUST be in the form of a Qualified
Name from a different namespace and are treated as
"unspecified".</xs:documentation>
 </xs:annotation>
 <xs:simpleType>
  <xs:union>
   <xs:simpleType>
    <xs:restriction base="xs:NCName">
     <xs:enumeration value="unspecified"/>
     <xs:enumeration value="allrightsreserved"/>
     <xs:enumeration value="prerelease"/>
     <xs:enumeration value="other"/>
    </xs:restriction>
   </xs:simpleType>
   <xs:simpleType>
    <xs:restriction base="xs:QName"/>
   </xs:simpleType>
  </xs:union>
 </xs:simpleType>
</xs:attribute>
```

complexType **metadataType**

diagram



namespace	http://www.udirector.net/schemas/2009/uits/1.1
children	**nonce** **Distributor** **Time** **ProductID** **AssetID** **TID** **UID** **Media** **URL** **PA** **Copyright** **Extra**
used by	element **UITS/metadata**

source

```
<xs:complexType name="metadataType">
 <xs:sequence>
  <xs:element name="nonce">
   <xs:annotation>
    <xs:documentation>The nonce is an 8 digit Base64 encoded random value
generated at the time of sale that is used to randomize the hash computation of the
signature value for security purposes.</xs:documentation>
   </xs:annotation>
   <xs:simpleType>
    <xs:restriction base="xs:string">
     <xs:length value="8"/>
    </xs:restriction>
   </xs:simpleType>
  </xs:element>
  <xs:element name="Distributor">
   <xs:annotation>
    <xs:documentation>This is the retailer or distributor's name or an associated
identifier in clear text.  Note that this does not identify the CDN or entity generating
and/or embedding the UITS payload, but rather the name of the company that has
the record of sale (or its agent) and issues sales reports back to the Content
Owner.</xs:documentation>
   </xs:annotation>
   <xs:complexType>
    <xs:simpleContent>
     <xs:extension base="xs:string">
      <xs:anyAttribute namespace="##other" processContents="lax"/>
     </xs:extension>
    </xs:simpleContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="Time">
   <xs:annotation>
    <xs:documentation>The date and time of the purchase (not download or
signature) in ISO 8601 format. </xs:documentation>
   </xs:annotation>
   <xs:complexType>
    <xs:simpleContent>
     <xs:extension base="xs:dateTime">
      <xs:anyAttribute namespace="##other" processContents="lax"/>
     </xs:extension>
    </xs:simpleContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="ProductID">
   <xs:annotation>
    <xs:documentation>The Product ID identifies the parent collection from which
the current item originates. If the entire collection has been purchased or completed
during the transaction, then this is indicated through the "completed" attribute. The
ProductID type is passed as an attribute.

Valid ProductID Types currently include "UPC" or "GRID".
```

For example, if a user purchases a single track from an album, the Product ID identifies the associated album and the completed attribute is set to “false”. If instead the user purchases the entire album, the Product ID still identifies the associated album, but the completed attribute is set to “true.” Some retailers support a “complete my album” sale, and for those transactions the completed attribute would also be set to true. Only when the entire collection is completed in the same transaction is the completed attribute set to true.</xs:documentation>

```
    </xs:annotation>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="type" use="required">
            <xs:simpleType>
              <xs:union>
                <xs:simpleType>
                  <xs:restriction base="xs:NCName">
                    <xs:enumeration value="UPC"/>
                    <xs:enumeration value="GRID"/>
                  </xs:restriction>
                </xs:simpleType>
                <xs:simpleType>
                  <xs:restriction base="xs:QName"/>
                </xs:simpleType>
              </xs:union>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="completed" type="xs:boolean"/>
          <xs:anyAttribute namespace="##other" processContents="lax"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="AssetID">
    <xs:annotation>
      <xs:documentation>
```

The Asset ID identifies an individual element of a product, such as a track or video. The AssetID type is passed as an attribute. Valid AssetID types currently include “ISRC”. Delimiters such as dashes MUST NOT be included.</xs:documentation>

```
    </xs:annotation>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:NMTOKEN">
          <xs:attribute name="type" use="required">
            <xs:simpleType>
              <xs:union>
                <xs:simpleType>
                  <xs:restriction base="xs:NCName">
                    <xs:enumeration value="ISRC"/>
                  </xs:restriction>
                </xs:simpleType>
                <xs:simpleType>
                  <xs:restriction base="xs:QName"/>
                </xs:simpleType>
              </xs:union>
            </xs:simpleType>
```

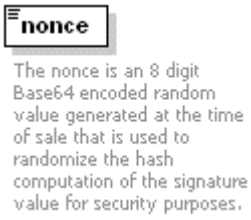
```
          </xs:attribute>
          <xs:anyAttribute namespace="##other" processContents="lax"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:choice maxOccurs="2">
    <xs:annotation>
      <xs:documentation>At least one of TID or UID is
REQUIRED.</xs:documentation>
    </xs:annotation>
    <xs:element name="TID">
      <xs:annotation>
        <xs:documentation>The transaction ID is the unique identifier for the
transaction.  Currently the only valid version is “1”.

Obfuscation MAY be used so long as the algorithm is deterministic, meaning the
same output is calculated given the same input. For example, a standard
cryptographic HMAC algorithm may be used. A list MUST be maintained of all issued
identifiers to prevent collisions and to assist with breach detection.
</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:string">
            <xs:attribute name="version" type="xs:unsignedInt" use="required">
              <xs:annotation>
                <xs:documentation>Currently, the only valid version is
"1".</xs:documentation>
              </xs:annotation>
            </xs:attribute>
            <xs:anyAttribute namespace="##other" processContents="lax"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="UID">
      <xs:annotation>
        <xs:documentation>The User ID is the unique identifier for the user.  Currently
the only valid version is “1”.

Obfuscation MAY be used so long as the algorithm is deterministic, meaning the
same output is calculated given the same input. For example, a standard
cryptographic HMAC algorithm may be used. A list MUST be maintained of all issued
identifiers to prevent collisions and to assist with breach detection.
</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:string">
            <xs:attribute name="version" type="xs:unsignedInt" use="required">
              <xs:annotation>
                <xs:documentation>Currently, the only valid version is
"1".</xs:documentation>
              </xs:annotation>
            </xs:attribute>
```

```
            <xs:anyAttribute namespace="##other" processContents="lax"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
  </xs:choice>
  <xs:element name="Media" minOccurs="0">
    <xs:annotation>
      <xs:documentation>A hash of the media (e.g. the audio, video, etc.) portion of
the file MUST be included, such that the UITS payload can be directly tied to the file
associated with it. The hash type is passed as an attribute, and the currently valid
type is “SHA256”.  Metadata fields MUST NOT be included in the hash computation,
because it must be possible for users to update standard ID3 or similar tags without
affecting the audio hash.  Note that the hash can be pre-computed so that it is not
necessary to compute it at the time of sale.

                                                    Instructions for MP3s:  The hash value for
the audio part of the MP3 is calculated using the hash algorithm over all of the audio
frames within the file in the order in which they appear in the file. The audio frames all
start with a 12-bit syncword, their frame size is calculated in the standard manner
from the bitrate, sampling and padding.  Frames that are not audio (such as frames
containing ID3 tags) are excluded.  In addition, one type of audio-related frame must
be identified and not included in the cryptographic hash, specifically, older Xing
variable byte rate data frames. They are documented at: http://gabriel.mp3-
tech.org/mp3infotag.html and http://www.codeproject.com/KB/audio-
video/mpegaudioinfo.aspx.

                                                    Note that future versions of UITS may
support fingerprints, embedded watermarks, or other techniques for identifying audio.
                                                    </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:base64Binary">
          <xs:attribute name="algorithm" use="required">
            <xs:annotation>
              <xs:documentation>The only defined type is "SHA256". Any other values, if
specified, must be in the form of a Qualified Name against a different
namespace.</xs:documentation>
            </xs:annotation>
            <xs:simpleType>
              <xs:union>
                <xs:simpleType>
                  <xs:restriction base="xs:NCName">
                    <xs:enumeration value="SHA256"/>
                  </xs:restriction>
                </xs:simpleType>
                <xs:simpleType>
                  <xs:restriction base="xs:QName"/>
                </xs:simpleType>
              </xs:union>
            </xs:simpleType>
          </xs:attribute>
          <xs:anyAttribute namespace="##other" processContents="lax"/>
        </xs:extension>
      </xs:simpleContent>
```

```
      </xs:complexType>
    </xs:element>
    <xs:element name="URL" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>The URL field is used for links that are worth transporting in
a cryptographically signed manner.  The type field specifies the URL type and
currently uses a subset of ID3v2.3 tag names.  Options for the type field include
WCOM, WCOP, WOAF, WOAR, WOAS, WORS, WPAY, and WPUB, which have the
meaning described in the ID3 specification.  In addition, a legal URL type is KeyURI,
which identifies the public key needed to validate the signature (see the
Implementation Details section below).  More than one URL element MAY be
included.

Applications may perform discovery on included URLs to identify associated services.
(See http://tools.ietf.org/html/draft-hammer-discovery).</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:anyURI">
            <xs:attribute name="type">
              <xs:simpleType>
                <xs:union>
                  <xs:simpleType>
                    <xs:restriction base="xs:NCName">
                      <xs:enumeration value="WCOM"/>
                      <xs:enumeration value="WCOP"/>
                      <xs:enumeration value="WOAF"/>
                      <xs:enumeration value="WOAR"/>
                      <xs:enumeration value="WOAS"/>
                      <xs:enumeration value="WORS"/>
                      <xs:enumeration value="WPAY"/>
                      <xs:enumeration value="WPUB"/>
                      <xs:enumeration value="KeyURI"/>
                    </xs:restriction>
                  </xs:simpleType>
                  <xs:simpleType>
                    <xs:restriction base="xs:QName"/>
                  </xs:simpleType>
                </xs:union>
              </xs:simpleType>
            </xs:attribute>
            <xs:anyAttribute namespace="##other" processContents="lax"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="PA" minOccurs="0">
      <xs:annotation>
        <xs:documentation>The Parental Advisory field is used to indicate the parental
advisory status for the track. The three valid values are “unspecified”, “explicit”, and
“edited”.  The absence of the PA element can be interpreted as
“unspecified”.</xs:documentation>
      </xs:annotation>
      <xs:simpleType>
        <xs:union>
          <xs:simpleType>
```

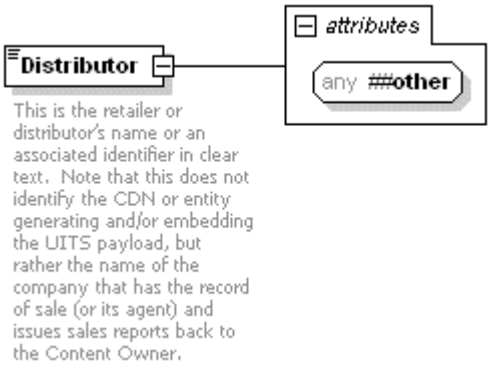
```
            <xs:restriction base="xs:NCName">
              <xs:enumeration value="unspecified"/>
              <xs:enumeration value="explicit"/>
              <xs:enumeration value="edited"/>
            </xs:restriction>
          </xs:simpleType>
          <xs:simpleType>
            <xs:restriction base="xs:QName"/>
          </xs:simpleType>
        </xs:union>
      </xs:simpleType>
    </xs:element>
    <xs:element name="Copyright" type="uits:copyrightType" minOccurs="0">
      <xs:annotation>
        <xs:documentation>This field can be used to communicate copyright information
about the associated track.  The value field can be one of these pre-defined values:
“unspecified”, “allrightsreserved ”, “prerelease ”, or “other”.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="Extra" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>The Extra field is used for miscellaneous metadata that is
worth transporting in a cryptographically signed manner.  The type field specifies the
type of metadata and the value field specifies the value. More than one Extra element
MAY be included.</xs:documentation>
      </xs:annotation>
      <xs:complexType mixed="true">
        <xs:sequence>
          <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="type" type="xs:Name"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

element **metadataType/nonce**

diagram	
type	restriction of **xs:string**
properties	isRef 0 content simple
facets	Kind Value annotation length 8
annotation	documentation The nonce is an 8 digit Base64 encoded random value generated at the time of sale that is used to

	randomize the hash computation of the signature value for security purposes.
source	<xs:element name="nonce"> <xs:annotation> <xs:documentation>The nonce is an 8 digit Base64 encoded random value generated at the time of sale that is used to randomize the hash computation of the signature value for security purposes.</xs:documentation> </xs:annotation> <xs:simpleType> <xs:restriction base="xs:string"> <xs:length value="8"/> </xs:restriction> </xs:simpleType> </xs:element>

element **metadataType/Distributor**

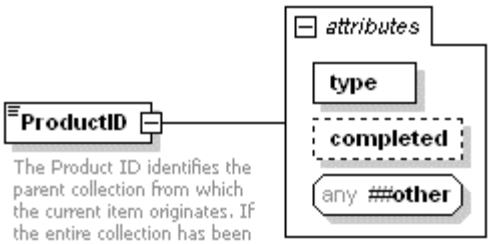
diagram	
type	extension of **xs:string**
properties	isRef 0 content complex
attributes	Name Type Use Default Fixed annotation
annotation	documentation This is the retailer or distributor's name or an associated identifier in clear text. Note that this does not identify the CDN or entity generating and/or embedding the UITS payload, but rather the name of the company that has the record of sale (or its agent) and issues sales reports back to the Content Owner.
source	<xs:element name="Distributor"> <xs:annotation> <xs:documentation>This is the retailer or distributor's name or an associated identifier in clear text. Note that this does not identify the CDN or entity generating and/or embedding the UITS payload, but rather the name of the company that has the record of sale (or its agent) and issues sales reports back to the Content Owner.</xs:documentation> </xs:annotation> <xs:complexType> <xs:simpleContent> <xs:extension base="xs:string"> <xs:anyAttribute namespace="##other" processContents="lax"/> </xs:extension> </xs:simpleContent> </xs:complexType> </xs:element>

element **metadataType/Time**

diagram	
type	extension of **xs:dateTime**
properties	isRef 0 content complex
attributes	Name Type Use Default Fixed annotation
annotation	documentation The date and time of the purchase (not download or signature) in ISO 8601 format.
source	<xs:element name="Time"> <xs:annotation> <xs:documentation>The date and time of the purchase (not download or signature) in ISO 8601 format. </xs:documentation> </xs:annotation> <xs:complexType> <xs:simpleContent> <xs:extension base="xs:dateTime"> <xs:anyAttribute namespace="##other" processContents="lax"/> </xs:extension> </xs:simpleContent> </xs:complexType> </xs:element>

element **metadataType/ProductID**

diagram



The Product ID identifies the parent collection from which the current item originates. If the entire collection has been purchased or completed during the transaction, then this is indicated through the "completed" attribute. The ProductID type is passed as an attribute.

Valid ProductID Types currently include "UPC" or "GRID".

For example, if a user purchases a single track from an album, the Product ID identifies the associated album and the completed attribute is set to "false". If instead the user purchases the entire album, the Product ID still identifies the associated album, but the completed attribute is set to "true." Some retailers support a "complete my album" sale, and for those transactions the completed attribute would also be set to true. Only when the entire collection is completed in the same transaction is the completed attribute set to true.

type: extension of **xs:string**

properties:

isRef	0
content	complex

attributes:

Name	Type	Use	Default	Fixed	annotation
type	**derived by: xs:NCName**	required			
completed	**xs:boolean**				

annotation:

documentation

The Product ID identifies the parent collection from which the current item originates. If the entire collection has been purchased or completed during the transaction, then this is indicated through the "completed" attribute. The ProductID type is passed as an attribute.

Valid ProductID Types currently include "UPC" or "GRID".

For example, if a user purchases a single track from an album, the Product ID identifies the associated album and the completed attribute is set to "false". If instead the user purchases the entire album, the Product ID still identifies the associated album, but the completed attribute is set to "true." Some retailers support a "complete my album" sale, and for those transactions the completed attribute would also be set to true. Only when the entire collection is completed in the same transaction is the completed attribute set to true.

source

```
<xs:element name="ProductID">
  <xs:annotation>
    <xs:documentation>The Product ID identifies the parent collection from which the
current item originates. If the entire collection has been purchased or completed during
the transaction, then this is indicated through the "completed" attribute. The ProductID
```

type is passed as an attribute.

Valid ProductID Types currently include “UPC” or “GRID”.

For example, if a user purchases a single track from an album, the Product ID identifies the associated album and the completed attribute is set to “false”. If instead the user purchases the entire album, the Product ID still identifies the associated album, but the completed attribute is set to “true.” Some retailers support a “complete my album” sale, and for those transactions the completed attribute would also be set to true. Only when the entire collection is completed in the same transaction is the completed attribute set to true.</xs:documentation>

```
  </xs:annotation>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="type" use="required">
          <xs:simpleType>
            <xs:union>
              <xs:simpleType>
                <xs:restriction base="xs:NCName">
                  <xs:enumeration value="UPC"/>
                  <xs:enumeration value="GRID"/>
                </xs:restriction>
              </xs:simpleType>
              <xs:simpleType>
                <xs:restriction base="xs:QName"/>
              </xs:simpleType>
            </xs:union>
          </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="completed" type="xs:boolean"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

attribute **metadataType/ProductID/@type**

type	union of (restriction of **xs:NCName**, **xs:QName**)
properties	isRef 0 use required
source	

```
<xs:attribute name="type" use="required">
  <xs:simpleType>
    <xs:union>
      <xs:simpleType>
        <xs:restriction base="xs:NCName">
          <xs:enumeration value="UPC"/>
          <xs:enumeration value="GRID"/>
        </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
        <xs:restriction base="xs:QName"/>
      </xs:simpleType>
```

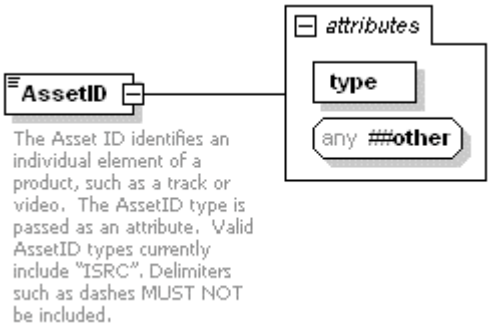
```
    </xs:union>
  </xs:simpleType>
</xs:attribute>
```

attribute **metadataType/ProductID/@completed**

type	**xs:boolean**
properties	isRef 0
source	`<xs:attribute name="completed" type="xs:boolean"/>`

element **metadataType/AssetID**

diagram



type	extension of **xs:NMTOKEN**
properties	isRef 0 content complex

attributes

Name	Type	Use	Default	Fixed	annotation
type	**derived by: xs:NCName**	required			

annotation

documentation
The Asset ID identifies an individual element of a product, such as a track or video. The AssetID type is passed as an attribute. Valid AssetID types currently include "ISRC". Delimiters such as dashes MUST NOT be included.

source

```
<xs:element name="AssetID">
  <xs:annotation>
    <xs:documentation>The Asset ID identifies an individual element of a product, such as a track or video.  The AssetID type is passed as an attribute.  Valid AssetID types currently include "ISRC". Delimiters such as dashes MUST NOT be included.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:NMTOKEN">
        <xs:attribute name="type" use="required">
          <xs:simpleType>
            <xs:union>
              <xs:simpleType>
                <xs:restriction base="xs:NCName">
                  <xs:enumeration value="ISRC"/>
                </xs:restriction>
              </xs:simpleType>
              <xs:simpleType>
                <xs:restriction base="xs:QName"/>
```

```
            </xs:simpleType>
          </xs:union>
        </xs:simpleType>
      </xs:attribute>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
</xs:element>
```

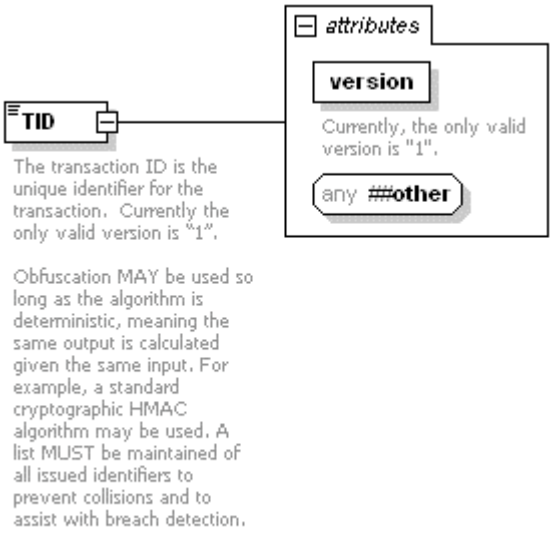
attribute **metadataType/AssetID/@type**

type	union of (restriction of **xs:NCName**, **xs:QName**)
properties	isRef 0 use required

```
<xs:attribute name="type" use="required">
 <xs:simpleType>
  <xs:union>
   <xs:simpleType>
    <xs:restriction base="xs:NCName">
     <xs:enumeration value="ISRC"/>
    </xs:restriction>
   </xs:simpleType>
   <xs:simpleType>
    <xs:restriction base="xs:QName"/>
   </xs:simpleType>
  </xs:union>
 </xs:simpleType>
</xs:attribute>
```

element **metadataType/TID**

diagram



type	extension of **xs:string**

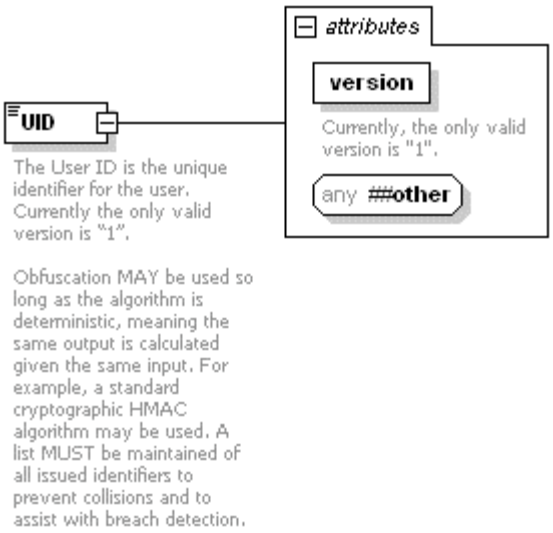
properties	isRef 0 content complex
attributes	Name: version Type: **xs:unsignedInt** Use: required Default: Fixed: annotation documentation Currently, the only valid version is "1".
annotation	documentation The transaction ID is the unique identifier for the transaction. Currently the only valid version is “1”. Obfuscation MAY be used so long as the algorithm is deterministic, meaning the same output is calculated given the same input. For example, a standard cryptographic HMAC algorithm may be used. A list MUST be maintained of all issued identifiers to prevent collisions and to assist with breach detection.
source	<xs:element name="TID"> <xs:annotation> <xs:documentation>The transaction ID is the unique identifier for the transaction. Currently the only valid version is “1”. Obfuscation MAY be used so long as the algorithm is deterministic, meaning the same output is calculated given the same input. For example, a standard cryptographic HMAC algorithm may be used. A list MUST be maintained of all issued identifiers to prevent collisions and to assist with breach detection. </xs:documentation> </xs:annotation> <xs:complexType> <xs:simpleContent> <xs:extension base="xs:string"> <xs:attribute name="version" type="xs:unsignedInt" use="required"> <xs:annotation> <xs:documentation>Currently, the only valid version is "1".</xs:documentation> </xs:annotation> </xs:attribute> <xs:anyAttribute namespace="##other" processContents="lax"/> </xs:extension> </xs:simpleContent> </xs:complexType> </xs:element>

attribute **metadataType/TID/@version**

type	**xs:unsignedInt**
properties	isRef 0 use required
annotation	documentation Currently, the only valid version is "1".
source	<xs:attribute name="version" type="xs:unsignedInt" use="required"> <xs:annotation> <xs:documentation>Currently, the only valid version is "1".</xs:documentation> </xs:annotation> </xs:attribute>

element **metadataType/UID**

diagram



type: extension of **xs:string**

properties: isRef 0; content complex

attributes

Name	Type	Use	Default	Fixed	annotation
version	**xs:unsignedInt**	required			documentation: Currently, the only valid version is "1".

annotation

documentation

The User ID is the unique identifier for the user. Currently the only valid version is “1”.

Obfuscation MAY be used so long as the algorithm is deterministic, meaning the same output is calculated given the same input. For example, a standard cryptographic HMAC algorithm may be used. A list MUST be maintained of all issued identifiers to prevent collisions and to assist with breach detection.

source

```
<xs:element name="UID">
  <xs:annotation>
    <xs:documentation>The User ID is the unique identifier for the user.  Currently the only valid version is “1”.

Obfuscation MAY be used so long as the algorithm is deterministic, meaning the same output is calculated given the same input. For example, a standard cryptographic HMAC algorithm may be used. A list MUST be maintained of all issued identifiers to prevent collisions and to assist with breach detection.
</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="version" type="xs:unsignedInt" use="required">
          <xs:annotation>
            <xs:documentation>Currently, the only valid version is "1".</xs:documentation>
          </xs:annotation>
        </xs:attribute>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
      </xs:extension>
```

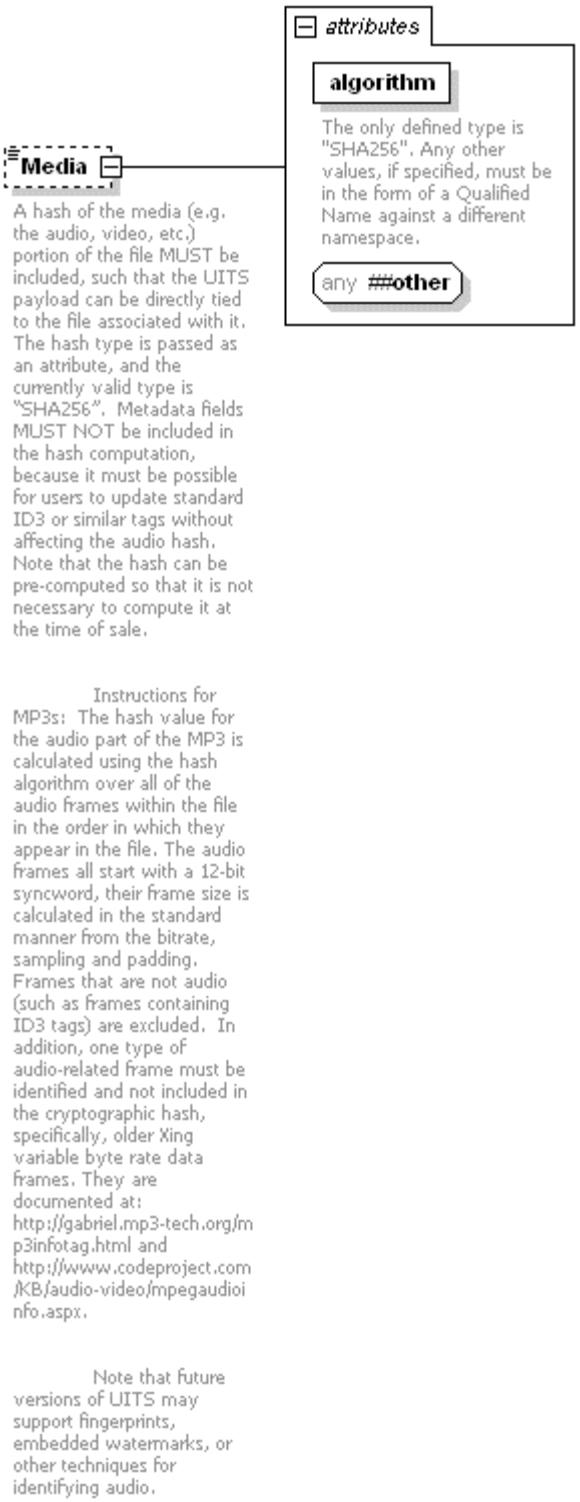
	`</xs:simpleContent>`
	`</xs:complexType>`
	`</xs:element>`

attribute **metadataType/UID/@version**

type	**xs:unsignedInt**
properties	isRef 0 use required
annotation	documentation Currently, the only valid version is "1".

source	`<xs:attribute name="version" type="xs:unsignedInt" use="required">`
	`<xs:annotation>`
	`<xs:documentation>Currently, the only valid version is "1".</xs:documentation>`
	`</xs:annotation>`
	`</xs:attribute>`

element **metadataType/Media**

diagram	
type	extension of **xs:base64Binary**
properties	isRef 0 minOcc 0

	maxOcc 1 content complex
attributes	Name: algorithm Type: **derived by: xs:NCName** Use: required Default: Fixed: annotation documentation The only defined type is "SHA256". Any other values, if specified, must be in the form of a Qualified Name against a different namespace.
annotation	documentation A hash of the media (e.g. the audio, video, etc.) portion of the file MUST be included, such that the UITS payload can be directly tied to the file associated with it. The hash type is passed as an attribute, and the currently valid type is “SHA256”. Metadata fields MUST NOT be included in the hash computation, because it must be possible for users to update standard ID3 or similar tags without affecting the audio hash. Note that the hash can be pre-computed so that it is not necessary to compute it at the time of sale. Instructions for MP3s: The hash value for the audio part of the MP3 is calculated using the hash algorithm over all of the audio frames within the file in the order in which they appear in the file. The audio frames all start with a 12-bit syncword, their frame size is calculated in the standard manner from the bitrate, sampling and padding. Frames that are not audio (such as frames containing ID3 tags) are excluded. In addition, one type of audio-related frame must be identified and not included in the cryptographic hash, specifically, older Xing variable byte rate data frames. They are documented at: http://gabriel.mp3-tech.org/mp3infotag.html and http://www.codeproject.com/KB/audio-video/mpegaudioinfo.aspx. Note that future versions of UITS may support fingerprints, embedded watermarks, or other techniques for identifying audio.
source	<xs:element name="Media" minOccurs="0"> <xs:annotation> <xs:documentation>A hash of the media (e.g. the audio, video, etc.) portion of the file MUST be included, such that the UITS payload can be directly tied to the file associated with it. The hash type is passed as an attribute, and the currently valid type is “SHA256”. Metadata fields MUST NOT be included in the hash computation, because it must be possible for users to update standard ID3 or similar tags without affecting the audio hash. Note that the hash can be pre-computed so that it is not necessary to compute it at the time of sale. Instructions for MP3s: The hash value for the audio part of the MP3 is calculated using the hash algorithm over all of the audio frames within the file in the order in which they appear in the file. The audio frames all start with a 12-bit syncword, their frame size is calculated in the standard manner from the bitrate, sampling and padding. Frames that are not audio (such as frames containing ID3 tags) are excluded. In addition, one type of audio-related frame must be identified and not included in the cryptographic hash, specifically, older Xing variable byte rate data frames. They are documented at: http://gabriel.mp3-tech.org/mp3infotag.html and http://www.codeproject.com/KB/audio-video/mpegaudioinfo.aspx. Note that future versions of UITS may support fingerprints, embedded watermarks, or other techniques for identifying audio. </xs:documentation> </xs:annotation> <xs:complexType>

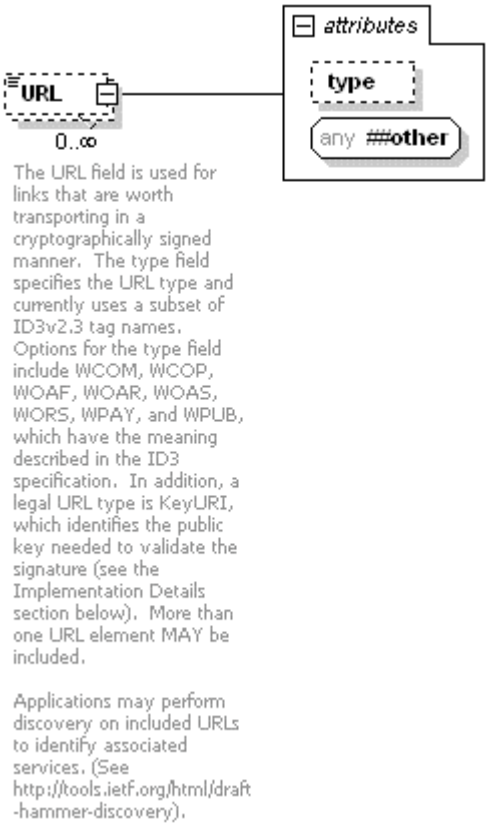
```
    <xs:simpleContent>
      <xs:extension base="xs:base64Binary">
        <xs:attribute name="algorithm" use="required">
          <xs:annotation>
            <xs:documentation>The only defined type is "SHA256". Any other values, if
specified, must be in the form of a Qualified Name against a different
namespace.</xs:documentation>
          </xs:annotation>
          <xs:simpleType>
            <xs:union>
              <xs:simpleType>
                <xs:restriction base="xs:NCName">
                  <xs:enumeration value="SHA256"/>
                </xs:restriction>
              </xs:simpleType>
              <xs:simpleType>
                <xs:restriction base="xs:QName"/>
              </xs:simpleType>
            </xs:union>
          </xs:simpleType>
        </xs:attribute>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

attribute **metadataType/Media/@algorithm**

type	union of (restriction of **xs:NCName**, **xs:QName**)
properties	isRef 0 use required
annotation	documentation The only defined type is "SHA256". Any other values, if specified, must be in the form of a Qualified Name against a different namespace.
source	

```
<xs:attribute name="algorithm" use="required">
  <xs:annotation>
    <xs:documentation>The only defined type is "SHA256". Any other values, if
specified, must be in the form of a Qualified Name against a different
namespace.</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:union>
      <xs:simpleType>
        <xs:restriction base="xs:NCName">
          <xs:enumeration value="SHA256"/>
        </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
        <xs:restriction base="xs:QName"/>
      </xs:simpleType>
    </xs:union>
  </xs:simpleType>
</xs:attribute>
```

element **metadataType/URL**

diagram	
type	extension of **xs:anyURI**
properties	isRef 0 minOcc 0 maxOcc unbounded content complex
attributes	Name: type Type: **derived by: xs:NCName** Use: Default: Fixed: annotation:
annotation	documentation The URL field is used for links that are worth transporting in a cryptographically signed manner. The type field specifies the URL type and currently uses a subset of ID3v2.3 tag names. Options for the type field include WCOM, WCOP, WOAF, WOAR, WOAS, WORS, WPAY, and WPUB, which have the meaning described in the ID3 specification. In addition, a legal URL type is KeyURI, which identifies the public key needed to validate the signature (see the Implementation Details section below). More than one URL element MAY be included. Applications may perform discovery on included URLs to identify associated services. (See http://tools.ietf.org/html/draft-hammer-discovery).
source	<xs:element name="URL" minOccurs="0" maxOccurs="unbounded"> <xs:annotation> <xs:documentation>The URL field is used for links that are worth transporting in a cryptographically signed manner. The type field specifies the URL type and currently uses a subset of ID3v2.3 tag names. Options for the type field include WCOM, WCOP, WOAF, WOAR, WOAS, WORS, WPAY, and WPUB, which have the meaning described in the ID3 specification. In addition, a legal URL type is KeyURI, which identifies the public key needed to validate the signature (see the Implementation Details section below). More than one URL element MAY be included.

```
Applications may perform discovery on included URLs to identify associated services.
(See http://tools.ietf.org/html/draft-hammer-discovery).</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:anyURI">
        <xs:attribute name="type">
          <xs:simpleType>
            <xs:union>
              <xs:simpleType>
                <xs:restriction base="xs:NCName">
                  <xs:enumeration value="WCOM"/>
                  <xs:enumeration value="WCOP"/>
                  <xs:enumeration value="WOAF"/>
                  <xs:enumeration value="WOAR"/>
                  <xs:enumeration value="WOAS"/>
                  <xs:enumeration value="WORS"/>
                  <xs:enumeration value="WPAY"/>
                  <xs:enumeration value="WPUB"/>
                  <xs:enumeration value="KeyURI"/>
                </xs:restriction>
              </xs:simpleType>
              <xs:simpleType>
                <xs:restriction base="xs:QName"/>
              </xs:simpleType>
            </xs:union>
          </xs:simpleType>
        </xs:attribute>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

attribute **metadataType/URL/@type**

type	union of (restriction of **xs:NCName**, **xs:QName**)
properties	isRef 0

```
source
<xs:attribute name="type">
  <xs:simpleType>
    <xs:union>
      <xs:simpleType>
        <xs:restriction base="xs:NCName">
          <xs:enumeration value="WCOM"/>
          <xs:enumeration value="WCOP"/>
          <xs:enumeration value="WOAF"/>
          <xs:enumeration value="WOAR"/>
          <xs:enumeration value="WOAS"/>
          <xs:enumeration value="WORS"/>
          <xs:enumeration value="WPAY"/>
          <xs:enumeration value="WPUB"/>
          <xs:enumeration value="KeyURI"/>
        </xs:restriction>
      </xs:simpleType>
```

```
      <xs:simpleType>
        <xs:restriction base="xs:QName"/>
      </xs:simpleType>
    </xs:union>
  </xs:simpleType>
</xs:attribute>
```

element **metadataType/PA**

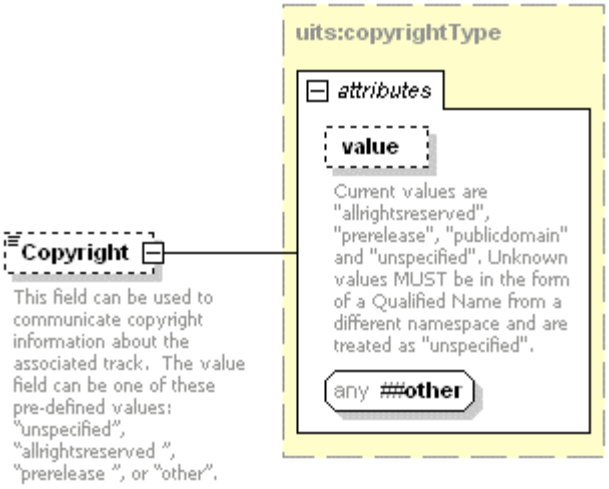
diagram	PA The Parental Advisory field is used to indicate the parental advisory status for the track. The three valid values are "unspecified", "explicit", and "edited". The absence of the PA element can be interpreted as "unspecified".
type	union of (restriction of **xs:NCName**, **xs:QName**)
properties	isRef 0 minOcc 0 maxOcc 1 content simple
annotation	documentation The Parental Advisory field is used to indicate the parental advisory status for the track. The three valid values are "unspecified", "explicit", and "edited". The absence of the PA element can be interpreted as "unspecified".

source

```
<xs:element name="PA" minOccurs="0">
  <xs:annotation>
    <xs:documentation>The Parental Advisory field is used to indicate the parental
advisory status for the track. The three valid values are "unspecified", "explicit", and
"edited". The absence of the PA element can be interpreted as
"unspecified".</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:union>
      <xs:simpleType>
        <xs:restriction base="xs:NCName">
          <xs:enumeration value="unspecified"/>
          <xs:enumeration value="explicit"/>
          <xs:enumeration value="edited"/>
        </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
        <xs:restriction base="xs:QName"/>
      </xs:simpleType>
    </xs:union>
  </xs:simpleType>
</xs:element>
```

element **metadataType/Copyright**

diagram



type: **uits:copyrightType**

properties

isRef	0
minOcc	0
maxOcc	1
content	complex
mixed	true

attributes

Name	Type	Use	Default	Fixed	annotation
value	**derived by: xs:NCName**				documentation Current values are "allrightsreserved", "prerelease", "publicdomain" and "unspecified". Unknown values MUST be in the form of a Qualified Name from a different namespace and are treated as "unspecified".

annotation

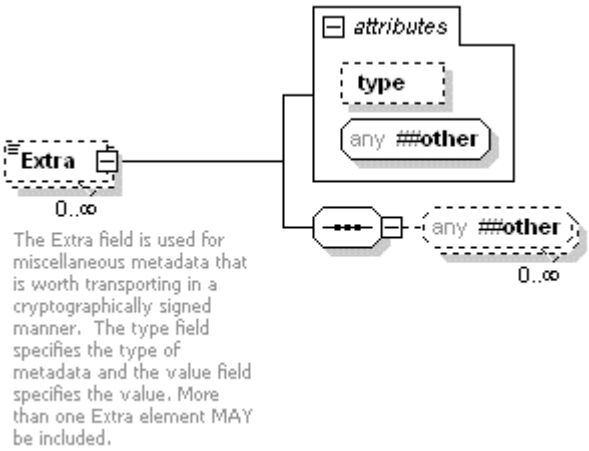
documentation

This field can be used to communicate copyright information about the associated track. The value field can be one of these pre-defined values: “unspecified”, “allrightsreserved ”, “prerelease ”, or “other”.

source

```
<xs:element name="Copyright" type="uits:copyrightType" minOccurs="0">
  <xs:annotation>
    <xs:documentation>This field can be used to communicate copyright information about the associated track.  The value field can be one of these pre-defined values: “unspecified”, “allrightsreserved ”, “prerelease ”, or “other”.</xs:documentation>
  </xs:annotation>
</xs:element>
```

element **metadataType/Extra**

diagram	
properties	isRef 0 minOcc 0 maxOcc unbounded content complex mixed true
attributes	Name: type; Type: **xs:Name**; Use; Default; Fixed; annotation
annotation	documentation The Extra field is used for miscellaneous metadata that is worth transporting in a cryptographically signed manner. The type field specifies the type of metadata and the value field specifies the value. More than one Extra element MAY be included.

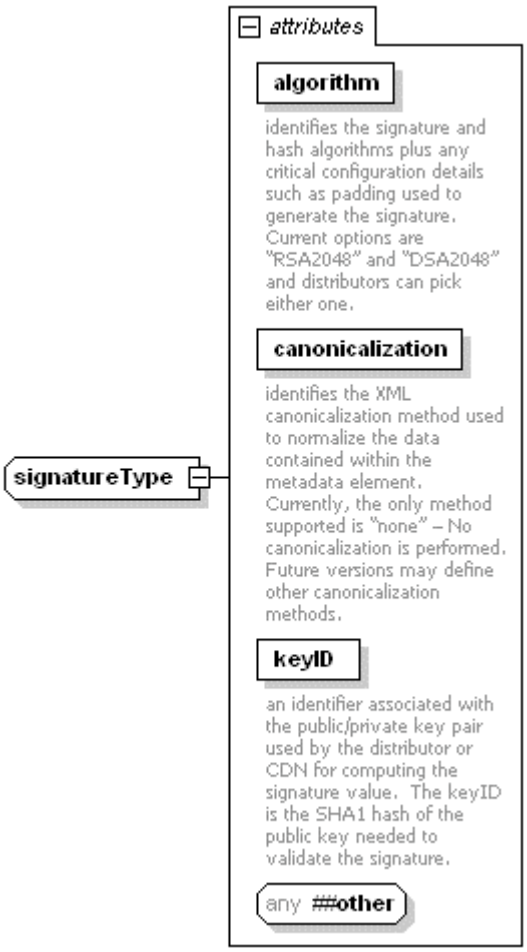
source	`<xs:element name="Extra" minOccurs="0" maxOccurs="unbounded">`
	`  <xs:annotation>`
	`  <xs:documentation>The Extra field is used for miscellaneous metadata that is worth transporting in a cryptographically signed manner. The type field specifies the type of metadata and the value field specifies the value. More than one Extra element MAY be included.</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:complexType mixed="true">`
	`  <xs:sequence>`
	`  <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>`
	`  </xs:sequence>`
	`  <xs:attribute name="type" type="xs:Name"/>`
	`  <xs:anyAttribute namespace="##other" processContents="lax"/>`
	`  </xs:complexType>`
	`</xs:element>`

attribute **metadataType/Extra/@type**

type	**xs:Name**
properties	isRef 0
source	`<xs:attribute name="type" type="xs:Name"/>`

complexType **signatureType**

diagram



namespace	http://www.udirector.net/schemas/2009/uits/1.1
type	extension of **xs:base64Binary**
properties	base xs:base64Binary
used by	element **UITS/signature**

attributes

Name	Type	Use	Default	Fixed	annotation
algorithm	**derived by: xs:NCName**	required			documentation identifies the signature and hash algorithms plus any critical configuration details such as padding used to generate the signature. Current options are “RSA2048” and “DSA2048” and distributors can pick either one.
canonicalization	**derived by: xs:NCName**	required			documentation identifies the XML canonicalization

	keyID	**xs:NMTOKEN**	required	method used to normalize the data contained within the metadata element. Currently, the only method supported is “none” – No canonicalization is performed. Future versions may define other canonicalization methods. documentation an identifier associated with the public/private key pair used by the distributor or CDN for computing the signature value. The keyID is the SHA1 hash of the public key needed to validate the signature.

source

```
<xs:complexType name="signatureType">
 <xs:simpleContent>
  <xs:extension base="xs:base64Binary">
   <xs:attribute name="algorithm" use="required">
    <xs:annotation>
     <xs:documentation>identifies the signature and hash algorithms plus any
critical configuration details such as padding used to generate the signature.  Current
options are “RSA2048” and “DSA2048” and distributors can pick either
one.</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
     <xs:union>
      <xs:simpleType>
       <xs:restriction base="xs:NCName">
        <xs:enumeration value="RSA2048"/>
        <xs:enumeration value="DSA2048"/>
       </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
       <xs:restriction base="xs:QName"/>
      </xs:simpleType>
     </xs:union>
    </xs:simpleType>
   </xs:attribute>
   <xs:attribute name="canonicalization" use="required">
    <xs:annotation>
     <xs:documentation>identifies the XML canonicalization method used to
normalize the data contained within the metadata element. Currently, the only method
```

```
supported is “none” – No canonicalization is performed. Future versions may define other canonicalization methods.</xs:documentation>
      </xs:annotation>
      <xs:simpleType>
        <xs:union>
          <xs:simpleType>
            <xs:restriction base="xs:NCName">
              <xs:enumeration value="none"/>
              <xs:enumeration value="clear_whitespace"/>
            </xs:restriction>
          </xs:simpleType>
          <xs:simpleType>
            <xs:restriction base="xs:QName"/>
          </xs:simpleType>
        </xs:union>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="keyID" type="xs:NMTOKEN" use="required">
      <xs:annotation>
        <xs:documentation>an identifier associated with the public/private key pair used by the distributor or CDN for computing the signature value.  The keyID is the SHA1 hash of the public key needed to validate the signature.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:extension>
 </xs:simpleContent>
</xs:complexType>
```

attribute **signatureType/@algorithm**

type	union of (restriction of **xs:NCName**, **xs:QName**)
properties	isRef 0 use required
annotation	documentation identifies the signature and hash algorithms plus any critical configuration details such as padding used to generate the signature. Current options are “RSA2048” and “DSA2048” and distributors can pick either one.
source	

```
<xs:attribute name="algorithm" use="required">
 <xs:annotation>
  <xs:documentation>identifies the signature and hash algorithms plus any critical configuration details such as padding used to generate the signature.  Current options are “RSA2048” and “DSA2048” and distributors can pick either one.</xs:documentation>
 </xs:annotation>
 <xs:simpleType>
  <xs:union>
   <xs:simpleType>
    <xs:restriction base="xs:NCName">
     <xs:enumeration value="RSA2048"/>
     <xs:enumeration value="DSA2048"/>
    </xs:restriction>
   </xs:simpleType>
   <xs:simpleType>
    <xs:restriction base="xs:QName"/>
```

```
      </xs:simpleType>
    </xs:union>
  </xs:simpleType>
</xs:attribute>
```

attribute **signatureType/@canonicalization**

type	union of (restriction of **xs:NCName**, **xs:QName**)
properties	isRef 0 use required
annotation	documentation identifies the XML canonicalization method used to normalize the data contained within the metadata element. Currently, the only method supported is “none” – No canonicalization is performed. Future versions may define other canonicalization methods.

source	`<xs:attribute name="canonicalization" use="required">`
	`  <xs:annotation>`
	`  <xs:documentation>identifies the XML canonicalization method used to normalize the data contained within the metadata element. Currently, the only method supported is “none” – No canonicalization is performed. Future versions may define other canonicalization methods.</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:simpleType>`
	`  <xs:union>`
	`  <xs:simpleType>`
	`  <xs:restriction base="xs:NCName">`
	`  <xs:enumeration value="none"/>`
	`  <xs:enumeration value="clear_whitespace"/>`
	`  </xs:restriction>`
	`  </xs:simpleType>`
	`  <xs:simpleType>`
	`  <xs:restriction base="xs:QName"/>`
	`  </xs:simpleType>`
	`  </xs:union>`
	`  </xs:simpleType>`
	`</xs:attribute>`

attribute **signatureType/@keyID**

type	**xs:NMTOKEN**
properties	isRef 0 use required
annotation	documentation an identifier associated with the public/private key pair used by the distributor or CDN for computing the signature value. The keyID is the SHA1 hash of the public key needed to validate the signature.

source	`<xs:attribute name="keyID" type="xs:NMTOKEN" use="required">`
	`  <xs:annotation>`
	`  <xs:documentation>an identifier associated with the public/private key pair used by the distributor or CDN for computing the signature value. The keyID is the SHA1 hash of the public key needed to validate the signature.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`