

Schema documentation for openSDX_00-00-00-01.xsd

august 16, 2011

Table of Contents

Namespace: ""	5
Schema(s)	5
Main schema openSDX_00-00-00-01.xsd	5
Element(s)	5
Element feed	5
Element feed / feedinfo	6
Element feedinfo / onlytest	7
Element feedinfo / feedid	7
Element feedinfo / creationdatetime	7
Element feedinfo / effectivedatetime	7
Element feedinfo / creator	7
Element creator / email	8
Element creator / userid	8
Element creator / keyid	8
Element feedinfo / receiver	8
Element receiver / type	9
Element receiver / servername	9
Element receiver / serveripv4	10
Element receiver / serveripv6	10
Element receiver / authtype	10
Element receiver / username	10
Element receiver / crypto	10
Element crypto / relatedemail	11
Element crypto / usedkeyid	11
Element crypto / usedpubkey	11
Element feedinfo / sender	11
Element sender / contractpartnerid	12
Element sender / ourcontractpartnerid	12
Element sender / email	12
Element sender / keyid	12
Element feedinfo / licensor	13
Element licensor / contractpartnerid	13
Element licensor / ourcontractpartnerid	13
Element licensor / email	14
Element licensor / keyid	14
Element feedinfo / licensee	14
Element licensee / contractpartnerid	14
Element licensee / ourcontractpartnerid	15
Element licensee / email	15
Element licensee / keyid	15
Element feedinfo / actions	15
Element actions / oninitialreceive	16
Element event / mailto	16
Element mailto / receiver	17
Element mailto / subject	17
Element mailto / text	17
Element event / http	17
Element http / url	18
Element http / type	18
Element http / addheader	18
Element http / addparams	19
Element event / fax	19
Element event / letter	19
Element letter / registered	20
Element letter / to	20
Element to / name	20
Element to / department	21
Element to / nameperson	21
Element to / street	21
Element to / postcode	21

Element to / country	22
Element to / additionaladdressinfo	22
Element letter / text	22
Element letter / costscoveredby	22
Element costscoveredby / contractpartnerid	23
Element costscoveredby / ourcontractpartnerid	23
Element costscoveredby / maxcostscovered	23
Element actions / onprocesstart	23
Element actions / onprocesend	24
Element actions / onfullsuccess	24
Element actions / onerror	25
Element feed / bundle	25
Element bundle / displayname	27
Element bundle / name	27
Element bundle / version	27
Element bundle / display_artistname	27
Element bundle / ids	27
Element ids / grid	28
Element ids / upc	29
Element ids / isrc	29
Element ids / contentauth	29
Element ids / labelordernum	29
Element ids / amzn	29
Element ids / isbn	30
Element ids / finetunes	30
Element ids / licensor	30
Element ids / licensee	30
Element ids / gvl	31
Element bundle / items	31
Element items / item	31
Element item / displayname	33
Element item / name	33
Element item / version	33
Element item / type	33
Element item / display_artistname	33
Element item / ids	34
Element item / contributors	34
Element contributors / contributor	35
Element contributor / name	35
Element contributor / type	36
Element contributor / year	36
Element contributor / ids	36
Element contributor / www	37
Element www / facebook	38
Element www / myspace	38
Element www / homepage	39
Element www / twitter	39
Element www / phone	40
Element item / information	40
Element information / texts	41
Element texts / promotext	42
Element texts / teasertext	42
Element information / physical_release_datetime	42
Element information / digital_release_datetime	43
Element information / playlength	43
Element information / num	43
Element information / setnum	43
Element information / suggested_prelistening_offset	44
Element information / origin_country	44
Element information / main_language	44
Element information / related	44
Element related / physical_distributor	45
Element related / utube	45
Element utube / url	46
Element utube / channel	46
Element related / bundle	46
Element bundle / contributors	48
Element bundle / information	48
Element bundle / license_basis	49
Element license_basis / territorial	49
Element territorial / territory	50
Element license_basis / timeframe	50

Element timeframe / from	51
Element timeframe / to	51
Element license_basis / pricing	51
Element pricing / pricecode	51
Element pricing / wholesale	52
Element license_basis / streaming_allowed	52
Element license_basis / channels	52
Element channels / channel	52
Element license_basis / as_on_bundle	53
Element bundle / license_specifics	53
Element license_specifics / rules	54
Element rules / rule	54
Element rule / if	54
Element if / what	55
Element if / operator	55
Element if / value	55
Element rule / then	56
Element then / echo	56
Element then / break	56
Element rule / else	56
Element else / proclaim	57
Element proclaim / what	57
Element proclaim / for	57
Element else / break	58
Element bundle / reporting	58
Element reporting / realtime	58
Element realtime / http	58
Element reporting / postponed	59
Element postponed / id	59
Element bundle / tags	60
Element tags / genres	60
Element genres / genre	60
Element tags / bundle_only	61
Element tags / explicit_lyrics	61
Element tags / live	61
Element tags / accoustic	61
Element tags / instrumental	62
Element bundle / files	62
Element files / file	62
Element file / location	64
Element fileLocation / path	64
Element fileLocation / http	64
Element fileHttp / url	65
Element fileHttp / user	65
Element fileHttp / pass	65
Element fileHttp / expiresdatetime	65
Element fileLocation / ftp	66
Element fileFtp / server	66
Element fileFtp / port	67
Element fileFtp / path	67
Element fileFtp / user	67
Element fileFtp / pass	67
Element fileFtp / expiresdatetime	67
Element file / type	68
Element file / filetype	68
Element file / samplerate	68
Element file / samplesize	68
Element file / bitrate	69
Element file / bitratetype	69
Element file / codec	69
Element file / codecsettings	69
Element file / bytes	70
Element file / checksums	70
Element checksums / md5	70
Element checksums / sha1	70
Element checksums / sha256	71
Element file / channels	71
Element file / dimension	71
Element dimension / width	72
Element dimension / height	72
Element file / decryptinfo	72
Element decryptinfo / cipher	73

Element decryptinfo / initvector	73
Element decryptinfo / key	73
Element decryptinfo / bytes	73
Element decryptinfo / checksums	73
Element bundle / purchase	74
Element purchase / pos	74
Element purchase / url	75
Element item / license_basis	75
Element item / license_specifics	75
Element item / tags	76
Element item / fingerprint	76
Element fingerprint / echoprint	77
Element item / reporting	77
Element item / files	77
Element feed / item	78
Complex Type(s)	79
Complex Type feedinfo	79
Complex Type creator	80
Complex Type receiver	80
Complex Type crypto	81
Complex Type sender	81
Complex Type licensor	82
Complex Type licensee	82
Complex Type actions	83
Complex Type event	83
Complex Type mailto	84
Complex Type action	84
Complex Type http	84
Complex Type http_addheader	85
Complex Type action_instruction	85
Complex Type http_addparams	85
Complex Type fax	86
Complex Type letter	86
Complex Type to	87
Complex Type costscoveredby	87
Complex Type bundle	88
Complex Type ids	89
Complex Type items	90
Complex Type item	90
Complex Type contributors	91
Complex Type contributor	92
Complex Type www	93
Complex Type publishable	93
Complex Type information	94
Complex Type texts	94
Complex Type promotext	95
Complex Type teasertext	95
Complex Type related	96
Complex Type physical_distributor	96
Complex Type utube	96
Complex Type license_basis	97
Complex Type territorial	97
Complex Type territory	98
Complex Type timeframe	98
Complex Type pricing	99
Complex Type channels	99
Complex Type channel	100
Complex Type license_specifics	100
Complex Type rules	100
Complex Type rule	101
Complex Type if	101
Complex Type then	102
Complex Type else	102
Complex Type proclaim	103
Complex Type reporting	103
Complex Type realtime	103
Complex Type postponed	104
Complex Type tags	104
Complex Type genres	105
Complex Type files	105
Complex Type file	105
Complex Type fileLocation	106

Complex Type fileHttp	107
Complex Type fileFtp	107
Complex Type checksums	108
Complex Type dimension	109
Complex Type decryptinfo	109
Complex Type purchase	109
Complex Type fingerprint	110
Complex Type oninitialreceive	110
Complex Type onprocessstart	111
Complex Type onprocessend	111
Complex Type onfullsuccess	112
Complex Type onerror	113
Simple Type(s)	113
Simple Type datetimeGMT	113
Simple Type email	113
Simple Type userid	114
Simple Type receivertypes	114
Simple Type iporhostname	114
Simple Type ipv4	114
Simple Type ipv6	115
Simple Type authtype	115
Simple Type keyid	115
Simple Type emaillist	116
Simple Type url	116
Simple Type httpmethods	116
Simple Type contributorType	116
Simple Type allowance	117
Simple Type operator	117
Simple Type explicitLyrics	118
Simple Type fileType	118
Simple Type fileChannels	118
Attribute(s)	119
Attribute publishable / @publishable	119
Attribute contributor / @num	119
Attribute promotext / @lang	119
Attribute teasertext / @lang	119
Attribute physical_distributor / @publishable	119
Attribute territory / @type	119
Attribute channel / @type	120
Attribute rule / @num	120

Namespace: ""

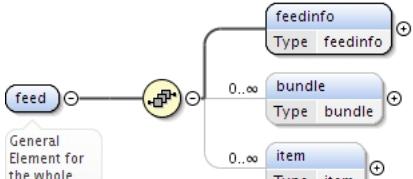
Schema(s)

Main schema openSDX_00-00-00-01.xsd

Namespace	No namespace
Properties	attribute form default: unqualified element form default: unqualified

Element(s)

Element feed

Namespace	No namespace
Annotations	General Element for the whole XML-Doc (root)
Diagram	 <pre> classDiagram class feed { <<General Element for the whole XML-Doc (root)>> } class feedinfo { <<Type feedinfo>> } class bundle { <<Type bundle>> } class item { <<Type item>> } feed "0..1" --> feedinfo feed "0..>" --> bundle feed "0..>" --> item </pre>
Properties	content: complex

Model	feedinfo , bundle* , item*
Children	bundle, feedinfo, item
Instance	<pre><feed> <feedinfo>{1,1}</feedinfo> <bundle>{0,unbounded}</bundle> <item>{0,unbounded}</item> </feed></pre>
Source	<pre><xsd:element name="feed"> <xsd:annotation> <xsd:documentation xml:lang="en">General Element for the whole XML-Doc (root)</ xsd:documentation> </xsd:annotation> <xsd:complexType> <xsd:sequence> <xsd:element name="feedinfo" type="feedinfo"/> <xsd:element name="bundle" type="bundle" maxOccurs="unbounded" minOccurs="0"/> <xsd:element name="item" type="item" maxOccurs="unbounded" minOccurs="0"/> </xsd:sequence> </xsd:complexType> </xsd:element></pre>

Element feed / feedinfo

Namespace	No namespace
Diagram	<pre> classDiagram class feedinfo { onlytest : xsd:boolean feedid : xsd:string creationdatetime : datetimeGMT effectivedatetime : datetimeGMT creator : creator receiver : receiver sender : sender licensor : licensor licensee : licensee actions : actions } note over feedinfo: On feedinfo-level there are the global information needed or at least valuable for ingesting / identifying the content... </pre>
Type	feedinfo
Properties	content: complex
Model	ALL(onlytest feedid creationdatetime effectivedatetime creator receiver sender licensor licensee actions{0,1})
Children	actions, creationdatetime, creator, effectivedatetime, feedid, licensee, licensor, onlytest, receiver, sender
Instance	<pre><feedinfo> <onlytest>{1,1}</onlytest> <feedid>{1,1}</feedid> <creationdatetime>{1,1}</creationdatetime> <effectivedatetime>{1,1}</effectivedatetime> <creator>{1,1}</creator> <receiver>{1,1}</receiver> <sender>{1,1}</sender> <licensor>{1,1}</licensor> <licensee>{1,1}</licensee></pre>

	<pre><actions>{0,1}</actions> </feedinfo></pre>
Source	<pre><xsd:element name="feedinfo" type="feedinfo" /></pre>

Element feedinfo / onlytest

Namespace	No namespace
Diagram	
Type	xsd:boolean
Properties	content: simple
Source	<pre><xsd:element name="onlytest" type="xsd:boolean" /></pre>

Element feedinfo / feedid

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple
Source	<pre><xsd:element name="feedid" type="xsd:string" /></pre>

Element feedinfo / creationdatetime

Namespace	No namespace
Diagram	
Type	datetimeGMT
Properties	content: simple
Facets	pattern $\d{4}-\d{2}-\d{2}$ $\d{2}:\d{2}:\d{2} \text{ GMT}$ $+\d{2}:\d{2}$
Source	<pre><xsd:element name="creationdatetime" type="datetimeGMT" /></pre>

Element feedinfo / effectivedatetime

Namespace	No namespace
Diagram	
Type	datetimeGMT
Properties	content: simple
Facets	pattern $\d{4}-\d{2}-\d{2}$ $\d{2}:\d{2}:\d{2} \text{ GMT}$ $+\d{2}:\d{2}$
Source	<pre><xsd:element name="effectivedatetime" type="datetimeGMT" /></pre>

Element feedinfo / creator

Namespace	No namespace
-----------	--------------

Diagram	<pre> classDiagram class creator { email userid keyid } creator < -- creator note over creator: This element contains information about the creator of that feed. </pre>
Type	creator
Properties	content: complex
Model	ALL(email userid keyid)
Children	email, keyid, userid
Instance	<pre> <creator> <email>{1,1}</email> <userid>{1,1}</userid> <keyid>{1,1}</keyid> </creator> </pre>
Source	<code><xsd:element name="creator" type="creator"/></code>

Element creator / email

Namespace	No namespace
Diagram	<pre> classDiagram class email { email } email < -- email </pre>
Type	email
Properties	content: simple
Source	<code><xsd:element name="email" type="email"/></code>

Element creator / userid

Namespace	No namespace
Diagram	<pre> classDiagram class userid { userid } userid < -- userid </pre>
Type	userid
Properties	content: simple
Source	<code><xsd:element name="userid" type="userid"/></code>

Element creator / keyid

Namespace	No namespace
Diagram	<pre> classDiagram class keyid { keyid } keyid < -- keyid note over keyid: Built-in primitive type. The string datatype represents character strings in XML. </pre>
Type	xsd:string
Properties	content: simple
Source	<code><xsd:element name="keyid" type="xsd:string"/></code>

Element feedinfo / receiver

Namespace	No namespace
-----------	--------------

Diagram	<pre> classDiagram receiver { type receivertypes servername iporhostname serveripv4 ipv4 serveripv6 ipv6 authtype username xsd:string crypto } receiver < -- receiver </pre> <p>This element contains information about the receiver of that feed.</p>
Type	receiver
Properties	content: complex
Model	ALL(type servername serveripv4 serveripv6 authtype username crypto)
Children	authtype, crypto, serveripv4, serveripv6, servername, type, username
Instance	<pre> <receiver> <type>{1,1}</type> <servername>{1,1}</servername> <serveripv4>{1,1}</serveripv4> <serveripv6>{1,1}</serveripv6> <authtype>{1,1}</authtype> <username>{1,1}</username> <crypto>{1,1}</crypto> </receiver> </pre>
Source	<code><xsd:element name="receiver" type="receiver" /></code>

Element receiver / type

Namespace	No namespace
Diagram	<pre> classDiagram type receivertypes type < -- receivertypes </pre>
Type	receivertypes
Properties	content: simple
Facets	enumeration ftp enumeration ftps enumeration sftp enumeration openSDX-Beam
Source	<code><xsd:element name="type" type="receivertypes" /></code>

Element receiver / servername

Namespace	No namespace
Diagram	<pre> classDiagram servername iporhostname servername < -- iporhostname </pre>
Type	iporhostname
Properties	content: simple
Source	<code><xsd:element name="servername" type="iporhostname" /></code>

Element receiver / serveripv4

Namespace	No namespace
Diagram	<pre> graph LR A[serveripv4] --- B[Type ipv4] A --- C[ipv4] </pre>
Type	ipv4
Properties	content: simple
Facets	pattern \d{3}\.\d{3}\.\d{3}\.\d{3}
Source	<xsd:element name="serveripv4" type="ipv4"/>

Element receiver / serveripv6

Namespace	No namespace
Diagram	<pre> graph LR A[serveripv6] --- B[Type ipv6] A --- C[ipv6] </pre>
Type	ipv6
Properties	content: simple
Source	<xsd:element name="serveripv6" type="ipv6"/>

Element receiver / authtype

Namespace	No namespace
Diagram	<pre> graph LR A[authtype] --- B[Type authtype] A --- C[authtype] </pre>
Type	authtype
Properties	content: simple
Facets	enumeration login enumeration keyfile enumeration kerberos enumeration keyfile+login enumeration keyfile+username
Source	<xsd:element name="authtype" type="authtype"/>

Element receiver / username

Namespace	No namespace
Diagram	<pre> graph LR A[username] --- B[Type xsd:string] A --- C[xsd:string] </pre> <p>Built-in primitive type. The string datatype represents character strings in XML.</p>
Type	xsd:string
Properties	content: simple
Source	<xsd:element name="username" type="xsd:string"/>

Element receiver / crypto

Namespace	No namespace
-----------	--------------

Diagram	<pre> classDiagram class crypto { <<crypto>> <<Type crypto>> } class relatedemail { <<relatedemail>> <<Type email>> } class usedkeyid { <<usedkeyid>> <<Type keyid>> } class usedpubkey { <<usedpubkey>> <<Type xsd:base64Binary>> } crypto "1..1" *--o relatedemail crypto "1..1" *--o usedkeyid crypto "1..1" *--o usedpubkey note over crypto: This element contains crypto information for secure and authenticated transfer. </pre>
Type	crypto
Properties	content: complex
Model	ALL(relatedemail usedkeyid usedpubkey)
Children	relatedemail, usedkeyid, usedpubkey
Instance	<pre> <crypto> <relatedemail>{1,1}</relatedemail> <usedkeyid>{1,1}</usedkeyid> <usedpubkey>{1,1}</usedpubkey> </crypto> </pre>
Source	<xsd:element name="crypto" type="crypto"/>

Element crypto / relatedemail

Namespace	No namespace
Diagram	<pre> classDiagram class relatedemail { <<relatedemail>> <<Type email>> } class email { <<email>> <<Type email>> } relatedemail "1..1" *--o email </pre>
Type	email
Properties	content: simple
Source	<xsd:element name="relatedemail" type="email"/>

Element crypto / usedkeyid

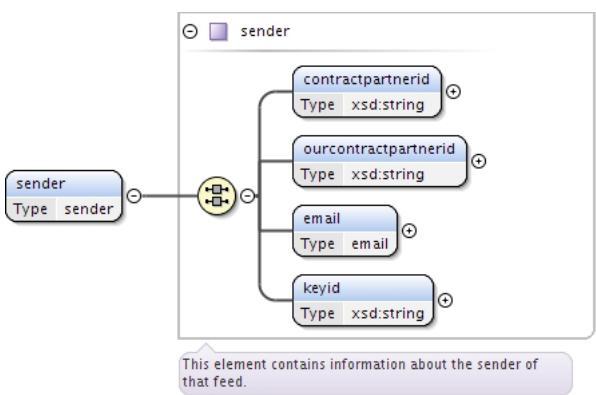
Namespace	No namespace
Diagram	<pre> classDiagram class usedkeyid { <<usedkeyid>> <<Type keyid>> } class keyid { <<keyid>> <<Type keyid>> } usedkeyid "1..1" *--o keyid </pre>
Type	keyid
Properties	content: simple
Source	<xsd:element name="usedkeyid" type="keyid"/>

Element crypto / usedpubkey

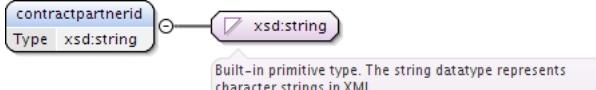
Namespace	No namespace
Diagram	<pre> classDiagram class usedpubkey { <<usedpubkey>> <<Type xsd:base64Binary>> } class xsdbase64Binary { <<xsd:base64Binary>> <<Type xsd:base64Binary>> } usedpubkey "1..1" *--o xsdbase64Binary </pre> <p>Built-in primitive type. The base64Binary datatype represents Base64-encoded arbitrary binary data.</p>
Type	xsd:base64Binary
Properties	content: simple
Source	<xsd:element name="usedpubkey" type="xsd:base64Binary"/>

Element feedinfo / sender

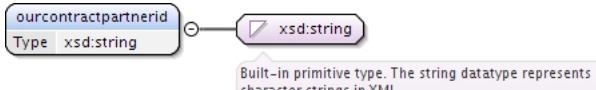
Namespace	No namespace
-----------	--------------

Diagram	
Type	sender
Properties	content: complex
Model	ALL(contractpartnerid ourcontractpartnerid email keyid)
Children	contractpartnerid, email, keyid, ourcontractpartnerid
Instance	<pre><sender> <contractpartnerid>{1,1}</contractpartnerid> <ourcontractpartnerid>{1,1}</ourcontractpartnerid> <email>{1,1}</email> <keyid>{1,1}</keyid> </sender></pre>
Source	<code><xsd:element name="sender" type="sender"/></code>

Element sender / contractpartnerid

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple
Source	<code><xsd:element name="contractpartnerid" type="xsd:string"/></code>

Element sender / ourcontractpartnerid

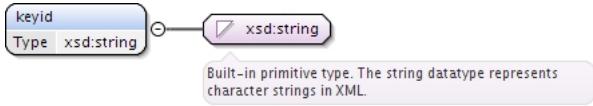
Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple
Source	<code><xsd:element name="ourcontractpartnerid" type="xsd:string"/></code>

Element sender / email

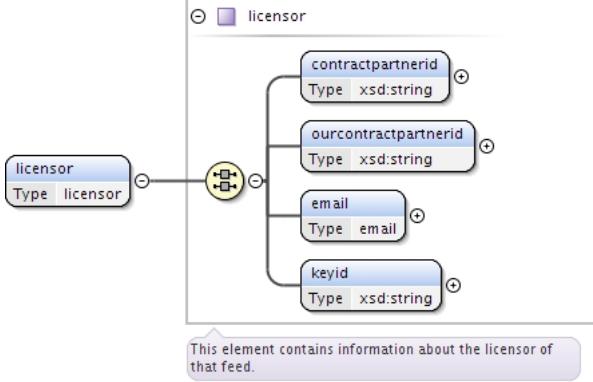
Namespace	No namespace
Diagram	
Type	email
Properties	content: simple
Source	<code><xsd:element name="email" type="email"/></code>

Element sender / keyid

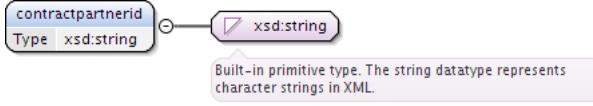
Namespace	No namespace
-----------	--------------

Diagram	 keyid Type xsd:string Built-in primitive type. The string datatype represents character strings in XML.
Type	xsd:string
Properties	content: simple
Source	<xsd:element name="keyid" type="xsd:string"/>

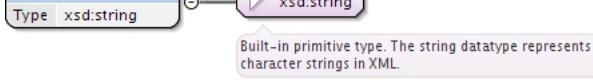
Element feedinfo / licensor

Namespace	No namespace
Diagram	 licensor Type licensor This element contains information about the licensor of that feed.
Type	licensor
Properties	content: complex
Model	ALL(contractpartnerid ourcontractpartnerid email keyid)
Children	contractpartnerid, email, keyid, ourcontractpartnerid
Instance	<licensor> <contractpartnerid>{1,1}</contractpartnerid> <ourcontractpartnerid>{1,1}</ourcontractpartnerid> <email>{1,1}</email> <keyid>{1,1}</keyid> </licensor>
Source	<xsd:element name="licensor" type="licensor"/>

Element licensor / contractpartnerid

Namespace	No namespace
Diagram	 contractpartnerid Type xsd:string Built-in primitive type. The string datatype represents character strings in XML.
Type	xsd:string
Properties	content: simple
Source	<xsd:element name="contractpartnerid" type="xsd:string"/>

Element licensor / ourcontractpartnerid

Namespace	No namespace
Diagram	 ourcontractpartnerid Type xsd:string Built-in primitive type. The string datatype represents character strings in XML.
Type	xsd:string
Properties	content: simple
Source	<xsd:element name="ourcontractpartnerid" type="xsd:string"/>

Element licensor / email

Namespace	No namespace
Diagram	<pre> graph LR email[Type email] --> email[/email] </pre>
Type	email
Properties	content: simple
Source	<xsd:element name="email" type="email"/>

Element licensor / keyid

Namespace	No namespace
Diagram	<pre> graph LR keyid[keyid Type xsd:string] --> xsdstring[xsd:string] </pre> <p>Built-in primitive type. The string datatype represents character strings in XML.</p>
Type	xsd:string
Properties	content: simple
Source	<xsd:element name="keyid" type="xsd:string"/>

Element feedinfo / licensee

Namespace	No namespace
Diagram	<pre> graph LR licensee[licensee Type licensee] --> licensee licensee --> contractpartnerid[contractpartnerid Type xsd:string] licensee --> ourcontractpartnerid[ourcontractpartnerid Type xsd:string] licensee --> email[email Type email] licensee --> keyid[keyid Type xsd:string] </pre> <p>This element contains information about the licensee of that feed.</p>
Type	licensee
Properties	content: complex
Model	ALL(contractpartnerid ourcontractpartnerid email keyid)
Children	contractpartnerid, email, keyid, ourcontractpartnerid
Instance	<licensee> <contractpartnerid>{1,1}</contractpartnerid> <ourcontractpartnerid>{1,1}</ourcontractpartnerid> <email>{1,1}</email> <keyid>{1,1}</keyid> </licensee>
Source	<xsd:element name="licensee" type="licensee"/>

Element licensee / contractpartnerid

Namespace	No namespace
Diagram	<pre> graph LR contractpartnerid[contractpartnerid Type xsd:string] --> xsdstring[xsd:string] </pre> <p>Built-in primitive type. The string datatype represents character strings in XML.</p>
Type	xsd:string
Properties	content: simple
Source	<xsd:element name="contractpartnerid" type="xsd:string"/>

Element licensee / ourcontractpartnerid

Namespace	No namespace
Diagram	<p>The diagram shows the 'ourcontractpartnerid' element with its type specified as 'xsd:string'. A tooltip indicates that 'xsd:string' is a built-in primitive type representing character strings in XML.</p>
Type	xsd:string
Properties	content: simple
Source	<xsd:element name="ourcontractpartnerid" type="xsd:string"/>

Element licensee / email

Namespace	No namespace
Diagram	<p>The diagram shows the 'email' element with its type specified as 'email'. A tooltip indicates that 'email' is a built-in primitive type.</p>
Type	email
Properties	content: simple
Source	<xsd:element name="email" type="email"/>

Element licensee / keyid

Namespace	No namespace
Diagram	<p>The diagram shows the 'keyid' element with its type specified as 'xsd:string'. A tooltip indicates that 'xsd:string' is a built-in primitive type representing character strings in XML.</p>
Type	xsd:string
Properties	content: simple
Source	<xsd:element name="keyid" type="xsd:string"/>

Element feedinfo / actions

Namespace	No namespace						
Diagram	<p>The diagram shows the 'actions' element, which contains five event types: 'oninitialreceive', 'onprocessstart', 'onprocessend', 'onfullsuccess', and 'onerror'. A tooltip indicates that this element contains information about possible actions with the feed.</p>						
Type	actions						
Properties	<table border="1"> <tr> <td>content:</td> <td>complex</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	complex	minOccurs:	0	maxOccurs:	1
content:	complex						
minOccurs:	0						
maxOccurs:	1						
Model	ALL(oninitialreceive onprocessstart onprocessend onfullsuccess onerror)						
Children	onerror, onfullsuccess, oninitialreceive, onprocessend, onprocessstart						
Instance	<actions>						

	<pre><oninitialreceive>{1,1}</oninitialreceive> <onprocessstart>{1,1}</onprocessstart> <onprocesend>{1,1}</onprocesend> <onfullsuccess>{1,1}</onfullsuccess> <onerror>{1,1}</onerror> </actions></pre>
Source	<pre><xsd:element name="actions" type="actions" maxOccurs="1" minOccurs="0" /></pre>

Element actions / oninitialreceive

Namespace	No namespace
Diagram	<p>This element contains information about possible events and actions.</p>
Type	event
Properties	content: complex
Model	mailto*, http*, fax*, letter*
Children	fax, http, letter, mailto
Instance	<pre><oninitialreceive> <mailto>{0,unbounded}</mailto> <http>{0,unbounded}</http> <fax>{0,unbounded}</fax> <letter>{0,unbounded}</letter> </oninitialreceive></pre>
Source	<pre><xsd:element name="oninitialreceive" type="event" /></pre>

Element event / mailto

Namespace	No namespace
Diagram	<p>This element contains information about mailto-event.</p>
Type	mailto
Type hierarchy	<ul style="list-style-type: none"> • action • mailto
Properties	<p>content: complex</p> <p>minOccurs: 0</p> <p>maxOccurs: unbounded</p>
Model	receiver+, subject, text
Children	receiver, subject, text

Instance	<pre><mailto> <receiver>{1,unbounded}</receiver> <subject>{1,1}</subject> <text>{1,1}</text> </mailto></pre>
Source	<pre><xsd:element name="mailto" type="mailto" minOccurs="0" maxOccurs="unbounded" /></pre>

Element mailto / receiver

Namespace	No namespace						
Diagram							
Type	emaillist						
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> <tr> <td>minOccurs:</td> <td>1</td> </tr> <tr> <td>maxOccurs:</td> <td>unbounded</td> </tr> </table>	content:	simple	minOccurs:	1	maxOccurs:	unbounded
content:	simple						
minOccurs:	1						
maxOccurs:	unbounded						
Source	<pre><xsd:element name="receiver" type="emaillist" minOccurs="1" maxOccurs="unbounded" /></pre>						

Element mailto / subject

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple
Source	<pre><xsd:element name="subject" type="xsd:string" /></pre>

Element mailto / text

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple
Source	<pre><xsd:element name="text" type="xsd:string" /></pre>

Element event / http

Namespace	No namespace
Diagram	

Type	http
Type hierarchy	<ul style="list-style-type: none"> • action <ul style="list-style-type: none"> • http
Properties	content: complex minOccurs: 0 maxOccurs: unbounded
Model	ALL(url type addheader addparams)
Children	addheader, addparams, type, url
Instance	<pre><http> <url>{1,1}</url> <type>{1,1}</type> <addheader>{1,1}</addheader> <addparams>{1,1}</addparams> </http></pre>
Source	<code><xsd:element name="http" type="http" minOccurs="0" maxOccurs="unbounded"/></code>

Element http / url

Namespace	No namespace
Diagram	
Type	url
Properties	content: simple
Source	<code><xsd:element name="url" type="url"/></code>

Element http / type

Namespace	No namespace
Diagram	
Type	httpmethods
Properties	content: simple
Facets	enumeration GET enumeration POST enumeration HEAD
Source	<code><xsd:element name="type" type="httpmethods"/></code>

Element http / addheader

Namespace	No namespace
Diagram	
Type	http_addheader
Type hierarchy	<ul style="list-style-type: none"> • action_instruction <ul style="list-style-type: none"> • http_addheader
Properties	content: complex
Model	ANY element from ANY namespace
Source	<code><xsd:element name="addheader" type="http_addheader"/></code>

Element http / addparams

Namespace	No namespace
Diagram	<pre> classDiagram class http_addparams { <<Base Type action_instruction>> } class addparams { <<Type http_addparams>> } http_addparams < -- addparams http_addparams "1..*" --> "#any" </pre>
Type	http_addparams
Type hierarchy	<ul style="list-style-type: none"> • action_instruction • http_addparams
Properties	content: complex
Model	ANY element from ANY namespace
Source	<xsd:element name="addparams" type="http_addparams" />

Element event / fax

Namespace	No namespace
Diagram	<pre> classDiagram class fax { <<Base Type action>> } class fax { <<Type fax>> } fax < -- fax fax "1..*" --> "#any" </pre>
Type	fax
Type hierarchy	<ul style="list-style-type: none"> • action • fax
Properties	<p>content: complex</p> <p>minOccurs: 0</p> <p>maxOccurs: unbounded</p>
Model	ANY element from ANY namespace
Source	<xsd:element name="fax" type="fax" minOccurs="0" maxOccurs="unbounded" />

Element event / letter

Namespace	No namespace
Diagram	<pre> classDiagram class letter { <<Base Type>> } class letter { <<Type letter>> } letter < -- registered letter < -- to letter < -- text letter < -- costscoveredby registered <--> letter to <--> letter text <--> letter costscoveredby <--> letter </pre> <p>This element contains information about the letter event.</p>
Type	letter
Properties	<p>content: complex</p> <p>minOccurs: 0</p> <p>maxOccurs: unbounded</p>

Model	ALL(registered to text costscoveredby)
Children	costscoveredby, registered, text, to
Instance	<pre><letter> <registered>{1,1}</registered> <to>{1,1}</to> <text>{1,1}</text> <costscoveredby>{1,1}</costscoveredby> </letter></pre>
Source	<code><xsd:element name="letter" type="letter" minOccurs="0" maxOccurs="unbounded" /></code>

Element letter / registered

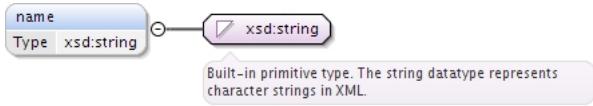
Namespace	No namespace
Diagram	
Type	xsd:boolean
Properties	content: simple
Source	<code><xsd:element name="registered" type="xsd:boolean" /></code>

Element letter / to

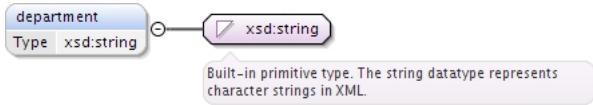
Namespace	No namespace
Diagram	
Type	to
Properties	content: complex
Model	ALL(name{0,1} department{0,1} nameperson{0,1} street postcode country additionaladdressinfo{0,1})
Children	additionaladdressinfo, country, department, name, nameperson, postcode, street
Instance	<pre><to> <name>{0,1}</name> <department>{0,1}</department> <nameperson>{0,1}</nameperson> <street>{1,1}</street> <postcode>{1,1}</postcode> <country>{1,1}</country> <additionaladdressinfo>{0,1}</additionaladdressinfo> </to></pre>
Source	<code><xsd:element name="to" type="to" /></code>

Element to / name

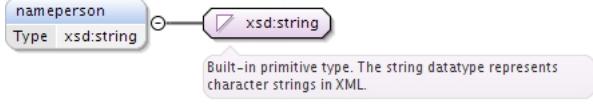
Namespace	No namespace
-----------	--------------

Diagram	
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="name" type="xsd:string" minOccurs="0" maxOccurs="1"/>

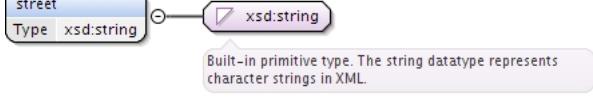
Element to / department

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="department" type="xsd:string" minOccurs="0" maxOccurs="1"/>

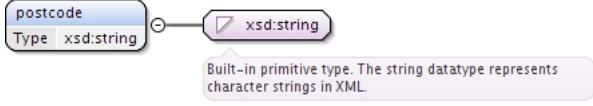
Element to / nameperson

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="nameperson" type="xsd:string" minOccurs="0" maxOccurs="1"/>

Element to / street

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple
Source	<xsd:element name="street" type="xsd:string" />

Element to / postcode

Namespace	No namespace
Diagram	
Type	xsd:string

Properties	content: simple
Source	<xsd:element name="postcode" type="xsd:string"/>

Element to / country

Namespace	No namespace
Diagram	<p>The diagram shows a blue rounded rectangle labeled "country" with a black border. To its right is a purple rounded rectangle labeled "xsd:string" with a black border. A line connects them with a small circle at the connection point. Below the diagram is a tooltip: "Built-in primitive type. The string datatype represents character strings in XML."</p>
Type	xsd:string
Properties	content: simple
Source	<xsd:element name="country" type="xsd:string"/>

Element to / additionaladdressinfo

Namespace	No namespace
Diagram	<p>The diagram shows a blue rounded rectangle labeled "additionaladdressinfo" with a black border. To its right is a purple rounded rectangle labeled "xsd:string" with a black border. A line connects them with a small circle at the connection point. Below the diagram is a tooltip: "Built-in primitive type. The string datatype represents character strings in XML."</p>
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="additionaladdressinfo" type="xsd:string" minOccurs="0" maxOccurs="1"/>

Element letter / text

Namespace	No namespace
Diagram	<p>The diagram shows a blue rounded rectangle labeled "text" with a black border. To its right is a purple rounded rectangle labeled "xsd:string" with a black border. A line connects them with a small circle at the connection point. Below the diagram is a tooltip: "Built-in primitive type. The string datatype represents character strings in XML."</p>
Type	xsd:string
Properties	content: simple
Source	<xsd:element name="text" type="xsd:string"/>

Element letter / costscoveredby

Namespace	No namespace
Diagram	<p>The diagram shows a blue rounded rectangle labeled "costscoveredby" with a black border. To its right is a purple rounded rectangle labeled "costscoveredby" with a black border. A line connects them with a small circle at the connection point. Inside the main "costscoveredby" box, there is another box containing three elements: "contractpartnerid" (xsd:string), "ourcontractpartnerid" (xsd:string), and "maxcostscovered" (xsd:string). Each of these three elements has a plus sign (+) next to it, indicating they are optional. Below the main box is a tooltip: "This element contains information about who covered the costs of event."</p>
Type	costscoveredby
Properties	content: complex
Model	ALL(contractpartnerid ourcontractpartnerid maxcostscovered{0,1})
Children	contractpartnerid, maxcostscovered, ourcontractpartnerid

Instance	<pre><costscoveredby> <contractpartnerid>{1,1}</contractpartnerid> <ourcontractpartnerid>{1,1}</ourcontractpartnerid> <maxcostscovered>{0,1}</maxcostscovered> </costscoveredby></pre>
Source	<pre><xsd:element name="costscoveredby" type="costscoveredby" /></pre>

Element costscoveredby / contractpartnerid

Namespace	No namespace
Diagram	<p>The diagram shows a box labeled "contractpartnerid" with a multiplicity of 0..1. An association line connects it to a box labeled "xsd:string". A callout bubble indicates that "xsd:string" is a "Built-in primitive type. The string datatype represents character strings in XML."</p>
Type	xsd:string
Properties	content: simple
Source	<pre><xsd:element name="contractpartnerid" type="xsd:string" /></pre>

Element costscoveredby / ourcontractpartnerid

Namespace	No namespace
Diagram	<p>The diagram shows a box labeled "ourcontractpartnerid" with a multiplicity of 0..1. An association line connects it to a box labeled "xsd:string". A callout bubble indicates that "xsd:string" is a "Built-in primitive type. The string datatype represents character strings in XML."</p>
Type	xsd:string
Properties	content: simple
Source	<pre><xsd:element name="ourcontractpartnerid" type="xsd:string" /></pre>

Element costscoveredby / maxcostscovered

Namespace	No namespace
Diagram	<p>The diagram shows a box labeled "maxcostscovered" with a multiplicity of 0..1. An association line connects it to a box labeled "xsd:string". A callout bubble indicates that "xsd:string" is a "Built-in primitive type. The string datatype represents character strings in XML."</p>
Type	xsd:string
Properties	<p>content: simple</p> <p>minOccurs: 0</p> <p>maxOccurs: 1</p>
Source	<pre><xsd:element name="maxcostscovered" type="xsd:string" minOccurs="0" maxOccurs="1" /></pre>

Element actions / onprocessstart

Namespace	No namespace
Diagram	<p>The diagram shows a box labeled "onprocessstart" with a multiplicity of 0..1. An association line connects it to a box labeled "event". Inside the "event" box, there are four boxes: "mailto" (multiplicity 0..oo), "http" (multiplicity 0..oo), "fax" (multiplicity 0..oo), and "letter" (multiplicity 0..oo). Each action has a "Type" field below it: "mailto", "http", "fax", and "letter". A callout bubble indicates that "This element contains information about possible events and actions."</p>

Type	event
Properties	content: complex
Model	mailto* , http* , fax* , letter*
Children	fax, http, letter, mailto
Instance	<pre><onprocesstart> <mailto>{0,unbounded}</mailto> <http>{0,unbounded}</http> <fax>{0,unbounded}</fax> <letter>{0,unbounded}</letter> </onprocesstart></pre>
Source	<xsd:element name="onprocesstart" type="event"/>

Element actions / onprocesend

Namespace	No namespace
Diagram	<p>This element contains information about possible events and actions.</p>
Type	event
Properties	content: complex
Model	mailto* , http* , fax* , letter*
Children	fax, http, letter, mailto
Instance	<pre><onprocesend> <mailto>{0,unbounded}</mailto> <http>{0,unbounded}</http> <fax>{0,unbounded}</fax> <letter>{0,unbounded}</letter> </onprocesend></pre>
Source	<xsd:element name="onprocesend" type="event"/>

Element actions / onfullsuccess

Namespace	No namespace
Diagram	<p>This element contains information about possible events and actions.</p>
Type	event
Properties	content: complex

Model	mailto*, http*, fax*, letter*
Children	fax, http, letter, mailto
Instance	<pre><onfullsuccess> <mailto>{0,unbounded}</mailto> <http>{0,unbounded}</http> <fax>{0,unbounded}</fax> <letter>{0,unbounded}</letter> </onfullsuccess></pre>
Source	<code><xsd:element name="onfullsuccess" type="event" /></code>

Element actions / onerror

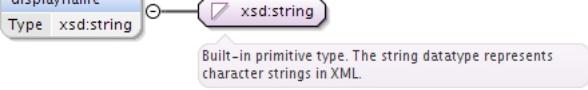
Namespace	No namespace
Diagram	<p>This element contains information about possible events and actions.</p>
Type	event
Properties	content: complex
Model	mailto*, http*, fax*, letter*
Children	fax, http, letter, mailto
Instance	<pre><onerror> <mailto>{0,unbounded}</mailto> <http>{0,unbounded}</http> <fax>{0,unbounded}</fax> <letter>{0,unbounded}</letter> </onerror></pre>
Source	<code><xsd:element name="onerror" type="event" /></code>

Element feed / bundle

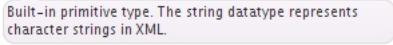
Namespace	No namespace
-----------	--------------

Diagram	<p>On bundle level, there are information on how to handle a collection of "items". This is mainly an album/ep/single. A...</p>						
Type	bundle						
Properties	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">content:</td><td style="padding: 2px;">complex</td></tr> <tr> <td style="padding: 2px;">minOccurs:</td><td style="padding: 2px;">0</td></tr> <tr> <td style="padding: 2px;">maxOccurs:</td><td style="padding: 2px;">unbounded</td></tr> </table>	content:	complex	minOccurs:	0	maxOccurs:	unbounded
content:	complex						
minOccurs:	0						
maxOccurs:	unbounded						
Model	ALL(displayname{0,1} name{0,1} version{0,1} display_artistname{0,1} ids items{0,1} contributors{0,1} information{0,1} license_basis{0,1} license_specifics{0,1} reporting{0,1} tags{0,1} files{0,1} purchase{0,1})						
Children	contributors, display_artistname, displayname, files, ids, information, items, license_basis, license_specifics, name, purchase, reporting, tags, version						
Instance	<pre><bundle> <displayname>{0,1}</displayname> <name>{0,1}</name> <version>{0,1}</version> <display_artistname>{0,1}</display_artistname> <ids>{1,1}</ids> <items>{0,1}</items> <contributors>{0,1}</contributors> <information>{0,1}</information> <license_basis>{0,1}</license_basis> <license_specifics>{0,1}</license_specifics> <reporting>{0,1}</reporting> <tags>{0,1}</tags> <files>{0,1}</files> <purchase>{0,1}</purchase> </bundle></pre>						
Source	<xsd:element name="bundle" type="bundle" maxOccurs="unbounded" minOccurs="0" />						

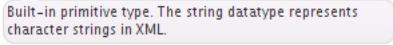
Element bundle / displayname

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<code><xsd:element name="displayname" type="xsd:string" maxOccurs="1" minOccurs="0" /></code>

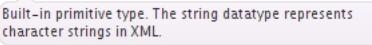
Element bundle / name

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<code><xsd:element name="name" type="xsd:string" maxOccurs="1" minOccurs="0" /></code>

Element bundle / version

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<code><xsd:element name="version" type="xsd:string" maxOccurs="1" minOccurs="0" /></code>

Element bundle / display_artistname

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<code><xsd:element name="display_artistname" type="xsd:string" maxOccurs="1" minOccurs="0" /></code>

Element bundle / ids

Namespace	No namespace
-----------	--------------

Diagram	
Type	ids
Properties	content: complex
Model	ALL(grid{0,1} upc{0,1} isrc{0,1} contentauth{0,1} labelordernum{0,1} amzn{0,1} isbn{0,1} finetunes{0,1} licensor{0,1} licensee{0,1} gvl{0,1})
Children	amzn, contentauth, finetunes, grid, gvl, isbn, isrc, labelordernum, licensee, licensor, upc
Instance	<pre><ids> <grid>{0,1}</grid> <upc>{0,1}</upc> <isrc>{0,1}</isrc> <contentauth>{0,1}</contentauth> <labelordernum>{0,1}</labelordernum> <amzn>{0,1}</amzn> <isbn>{0,1}</isbn> <finetunes>{0,1}</finetunes> <licensor>{0,1}</licensor> <licensee>{0,1}</licensee> <gvl>{0,1}</gvl> </ids></pre>
Source	<code><xsd:element name="ids" type="ids" /></code>

Element ids / grid

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<code><xsd:element name="grid" type="xsd:string" maxOccurs="1" minOccurs="0" /></code>

Element `ids / upc`

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<code><xsd:element name="upc" type="xsd:string" maxOccurs="1" minOccurs="0" /></code>

Element `ids / isrc`

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<code><xsd:element name="isrc" type="xsd:string" maxOccurs="1" minOccurs="0" /></code>

Element `ids / contentauth`

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<code><xsd:element name="contentauth" type="xsd:string" maxOccurs="1" minOccurs="0" /></code>

Element `ids / labelordernum`

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<code><xsd:element name="labelordernum" type="xsd:string" maxOccurs="1" minOccurs="0" /></code>

Element `ids / amzn`

Namespace	No namespace
-----------	--------------

Diagram	The diagram shows a blue rounded rectangle labeled "amzn" with a small "Type" label below it. A line connects it to a purple rounded rectangle labeled "xsd:string". A tooltip below the connection states: "Built-in primitive type. The string datatype represents character strings in XML."/>						
Type	xsd:string						
Properties	<table> <tr><td>content:</td><td>simple</td></tr> <tr><td>minOccurs:</td><td>0</td></tr> <tr><td>maxOccurs:</td><td>1</td></tr> </table>	content:	simple	minOccurs:	0	maxOccurs:	1
content:	simple						
minOccurs:	0						
maxOccurs:	1						
Source	<xsd:element name="amzn" type="xsd:string" maxOccurs="1" minOccurs="0" />						

Element ids / isbn

Namespace	No namespace						
Diagram	The diagram shows a blue rounded rectangle labeled "isbn" with a small "Type" label below it. A line connects it to a purple rounded rectangle labeled "xsd:string". A tooltip below the connection states: "Built-in primitive type. The string datatype represents character strings in XML."/>						
Type	xsd:string						
Properties	<table> <tr><td>content:</td><td>simple</td></tr> <tr><td>minOccurs:</td><td>0</td></tr> <tr><td>maxOccurs:</td><td>1</td></tr> </table>	content:	simple	minOccurs:	0	maxOccurs:	1
content:	simple						
minOccurs:	0						
maxOccurs:	1						
Source	<xsd:element name="isbn" type="xsd:string" maxOccurs="1" minOccurs="0" />						

Element ids / finetunes

Namespace	No namespace						
Diagram	The diagram shows a blue rounded rectangle labeled "finetunes" with a small "Type" label below it. A line connects it to a purple rounded rectangle labeled "xsd:string". A tooltip below the connection states: "Built-in primitive type. The string datatype represents character strings in XML."/>						
Type	xsd:string						
Properties	<table> <tr><td>content:</td><td>simple</td></tr> <tr><td>minOccurs:</td><td>0</td></tr> <tr><td>maxOccurs:</td><td>1</td></tr> </table>	content:	simple	minOccurs:	0	maxOccurs:	1
content:	simple						
minOccurs:	0						
maxOccurs:	1						
Source	<xsd:element name="finetunes" type="xsd:string" maxOccurs="1" minOccurs="0" />						

Element ids / licensor

Namespace	No namespace						
Diagram	The diagram shows a blue rounded rectangle labeled "licensor" with a small "Type" label below it. A line connects it to a purple rounded rectangle labeled "xsd:string". A tooltip below the connection states: "Built-in primitive type. The string datatype represents character strings in XML."/>						
Type	xsd:string						
Properties	<table> <tr><td>content:</td><td>simple</td></tr> <tr><td>minOccurs:</td><td>0</td></tr> <tr><td>maxOccurs:</td><td>1</td></tr> </table>	content:	simple	minOccurs:	0	maxOccurs:	1
content:	simple						
minOccurs:	0						
maxOccurs:	1						
Source	<xsd:element name="licensor" type="xsd:string" maxOccurs="1" minOccurs="0" />						

Element ids / licensee

Namespace	No namespace
-----------	--------------

Diagram	A diagram showing the 'licensee' element. It has a blue rounded rectangle labeled 'licensee' with a 'Type' tab showing 'xsd:string'. A line connects it to a purple rounded rectangle labeled 'xsd:string'. A tooltip below says: 'Built-in primitive type. The string datatype represents character strings in XML.'						
Type	xsd:string						
Properties	<table border="1"> <tr><td>content:</td><td>simple</td></tr> <tr><td>minOccurs:</td><td>0</td></tr> <tr><td>maxOccurs:</td><td>1</td></tr> </table>	content:	simple	minOccurs:	0	maxOccurs:	1
content:	simple						
minOccurs:	0						
maxOccurs:	1						
Source	<xsd:element name="licensee" type="xsd:string" maxOccurs="1" minOccurs="0"/>						

Element ids / gvl

Namespace	No namespace						
Diagram	A diagram showing the 'gvl' element. It has a blue rounded rectangle labeled 'gvl' with a 'Type' tab showing 'xsd:string'. A line connects it to a purple rounded rectangle labeled 'xsd:string'. A tooltip below says: 'Built-in primitive type. The string datatype represents character strings in XML.'						
Type	xsd:string						
Properties	<table border="1"> <tr><td>content:</td><td>simple</td></tr> <tr><td>minOccurs:</td><td>0</td></tr> <tr><td>maxOccurs:</td><td>1</td></tr> </table>	content:	simple	minOccurs:	0	maxOccurs:	1
content:	simple						
minOccurs:	0						
maxOccurs:	1						
Source	<xsd:element name="gvl" type="xsd:string" maxOccurs="1" minOccurs="0"/>						

Element bundle / items

Namespace	No namespace						
Diagram	A diagram showing the 'items' element. It has a blue rounded rectangle labeled 'items' with a 'Type' tab showing 'items'. A line connects it to another blue rounded rectangle labeled 'items'. Below them is a purple rounded rectangle labeled 'item' with a 'Type' tab showing 'item'. A line connects the first 'items' to the 'item' with a multiplicity '1..∞'. A tooltip below says: 'This element is a container for item-elements.'						
Type	items						
Properties	<table border="1"> <tr><td>content:</td><td>complex</td></tr> <tr><td>minOccurs:</td><td>0</td></tr> <tr><td>maxOccurs:</td><td>1</td></tr> </table>	content:	complex	minOccurs:	0	maxOccurs:	1
content:	complex						
minOccurs:	0						
maxOccurs:	1						
Model	item+						
Children	item						
Instance	<items> <item>{1,unbounded}</item> </items>						
Source	<xsd:element name="items" type="items" maxOccurs="1" minOccurs="0"/>						

Element items / item

Namespace	No namespace
-----------	--------------

Diagram	
	<pre> classDiagram class item { displayname name version type display_artistname ids contributors information license_basis license_specifics tags fingerprint reporting files } item < -- item note over item: This element contains information about a item just like a track. The type describes what the item is e.g. audio, ... </pre>
Type	item
Properties	<p>content: complex</p> <p>minOccurs: 1</p> <p>maxOccurs: unbounded</p>
Model	ALL(displayname name version type display_artistname{0,1} ids{0,1} contributors information license_basis license_specifics tags{0,1} fingerprint{0,1} reporting{0,1} files{0,1})
Children	contributors, display_artistname, displayname, files, fingerprint, ids, information, license_basis, license_specifics, name, reporting, tags, type, version
Instance	<pre> <item> <displayname>{1,1}</displayname> <name>{1,1}</name> <version>{1,1}</version> <type>{1,1}</type> <display_artistname>{0,1}</display_artistname> <ids>{0,1}</ids> <contributors>{1,1}</contributors> <information>{1,1}</information> <license_basis>{1,1}</license_basis> <license_specifics>{1,1}</license_specifics> <tags>{0,1}</tags> <fingerprint>{0,1}</fingerprint> <reporting>{0,1}</reporting> <files>{0,1}</files> </item> </pre>
Source	<xsd:element name="item" type="item" maxOccurs="unbounded" minOccurs="1"/>

Element item / displayname

Namespace	No namespace
Diagram	<p>The diagram shows the element 'displayname' highlighted in blue. A line connects it to a purple rounded rectangle containing the text 'xsd:string'. A tooltip below the connection states: 'Built-in primitive type. The string datatype represents character strings in XML.'</p>
Type	xsd:string
Properties	content: simple
Source	<xsd:element name="displayname" type="xsd:string"/>

Element item / name

Namespace	No namespace
Diagram	<p>The diagram shows the element 'name' highlighted in blue. A line connects it to a purple rounded rectangle containing the text 'xsd:string'. A tooltip below the connection states: 'Built-in primitive type. The string datatype represents character strings in XML.'</p>
Type	xsd:string
Properties	content: simple
Source	<xsd:element name="name" type="xsd:string"/>

Element item / version

Namespace	No namespace
Diagram	<p>The diagram shows the element 'version' highlighted in blue. A line connects it to a purple rounded rectangle containing the text 'xsd:string'. A tooltip below the connection states: 'Built-in primitive type. The string datatype represents character strings in XML.'</p>
Type	xsd:string
Properties	content: simple
Source	<xsd:element name="version" type="xsd:string"/>

Element item / type

Namespace	No namespace
Diagram	<p>The diagram shows the element 'type' highlighted in blue. A line connects it to a purple rounded rectangle containing the text 'xsd:string'. A tooltip below the connection states: 'Built-in primitive type. The string datatype represents character strings in XML.'</p>
Type	xsd:string
Properties	content: simple
Source	<xsd:element name="type" type="xsd:string"/>

Element item / display_artistname

Namespace	No namespace
Diagram	<p>The diagram shows the element 'display_artistname' highlighted in blue. A line connects it to a purple rounded rectangle containing the text 'xsd:string'. A tooltip below the connection states: 'Built-in primitive type. The string datatype represents character strings in XML.'</p>
Type	xsd:string
Properties	content: simple

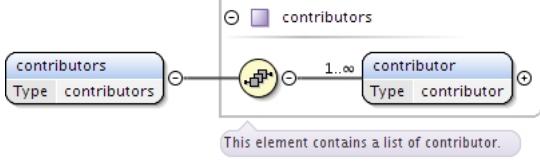
	minOccurs: 0
	maxOccurs: 1
Source	<xsd:element name="display_artistname" type="xsd:string" maxOccurs="1" minOccurs="0" />

Element item / ids

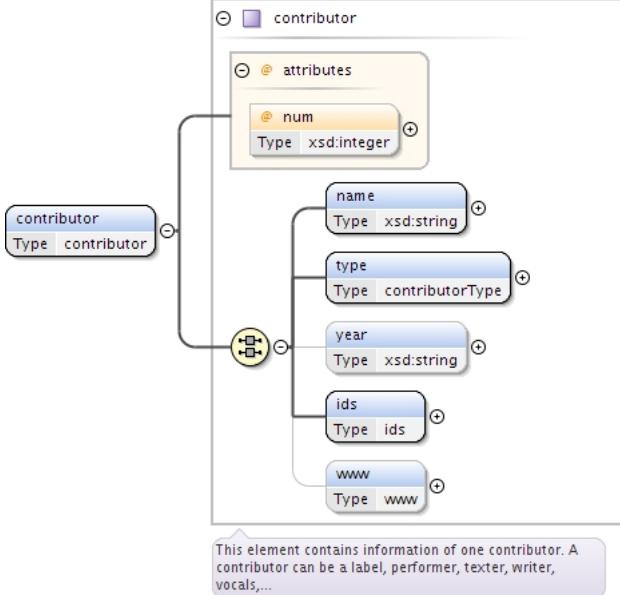
Namespace	No namespace
Diagram	<p>This Element is a container for all IDs which are available for the associated element.</p>
Type	ids
Properties	content: complex minOccurs: 0 maxOccurs: 1
Model	ALL(grid{0,1} upc{0,1} isrc{0,1} contentauth{0,1} labelordernum{0,1} amzn{0,1} isbn{0,1} finetunes{0,1} licensor{0,1} licensee{0,1} gvl{0,1})
Children	amzn, contentauth, finetunes, grid, gvl, isbn, isrc, labelordernum, licensee, licensor, upc
Instance	<ids> <grid>{0,1}</grid> <upc>{0,1}</upc> <isrc>{0,1}</isrc> <contentauth>{0,1}</contentauth> <labelordernum>{0,1}</labelordernum> <amzn>{0,1}</amzn> <isbn>{0,1}</isbn> <finetunes>{0,1}</finetunes> <licensor>{0,1}</licensor> <licensee>{0,1}</licensee> <gvl>{0,1}</gvl> </ids>
Source	<xsd:element name="ids" type="ids" maxOccurs="1" minOccurs="0" />

Element item / contributors

Namespace	No namespace
-----------	--------------

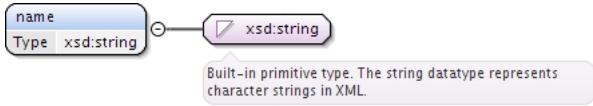
Diagram	 This element contains a list of contributor.
Type	contributors
Properties	content: complex
Model	contributor+
Children	contributor
Instance	<contributors> <contributor num="">{1,unbounded}</contributor> </contributors>
Source	<xsd:element name="contributors" type="contributors"/>

Element contributors / contributor

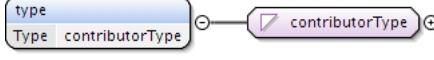
Namespace	No namespace										
Diagram	 This element contains information of one contributor. A contributor can be a label, performer, texter, writer, vocals,...										
Type	contributor										
Properties	content: complex minOccurs: 1 maxOccurs: unbounded										
Model	ALL(name type year{0,1} ids www{0,1})										
Children	ids, name, type, www, year										
Instance	<contributor num=""> <name>{1,1}</name> <type>{1,1}</type> <year>{0,1}</year> <ids>{1,1}</ids> <www>{0,1}</www> </contributor>										
Attributes	<table border="1"> <thead> <tr> <th>QName</th> <th>Type</th> <th>Fixed</th> <th>Default</th> <th>Use</th> </tr> </thead> <tbody> <tr> <td>num</td> <td>xsd:integer</td> <td></td> <td></td> <td>optional</td> </tr> </tbody> </table>	QName	Type	Fixed	Default	Use	num	xsd:integer			optional
QName	Type	Fixed	Default	Use							
num	xsd:integer			optional							
Source	<xsd:element name="contributor" type="contributor" maxOccurs="unbounded" minOccurs="1"/>										

Element contributor / name

Namespace	No namespace
-----------	--------------

Diagram	
Type	xsd:string
Properties	content: simple
Source	<xsd:element name="name" type="xsd:string" />

Element contributor / type

Namespace	No namespace
Diagram	
Type	contributorType
Properties	content: simple
Facets	enumeration label enumeration performer enumeration texter enumeration writer enumeration vocals enumeration conductor enumeration display_artist enumeration compilator enumeration copyright enumeration production enumeration clearinghouse
Source	<xsd:element name="type" type="contributorType" />

Element contributor / year

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="year" type="xsd:string" maxOccurs="1" minOccurs="0" />

Element contributor / ids

Namespace	No namespace
-----------	--------------

Diagram	<p>This Element is a container for all IDs which are available for the associated element.</p>
Type	ids
Properties	content: complex
Model	ALL(grid{0,1} upc{0,1} isrc{0,1} contentauth{0,1} labelordernum{0,1} amzn{0,1} isbn{0,1} finetunes{0,1} licensor{0,1} licensee{0,1} gvl{0,1})
Children	amzn, contentauth, finetunes, grid, gvl, isbn, isrc, labelordernum, licensee, licensor, upc
Instance	<pre><ids> <grid>{0,1}</grid> <upc>{0,1}</upc> <isrc>{0,1}</isrc> <contentauth>{0,1}</contentauth> <labelordernum>{0,1}</labelordernum> <amzn>{0,1}</amzn> <isbn>{0,1}</isbn> <finetunes>{0,1}</finetunes> <licensor>{0,1}</licensor> <licensee>{0,1}</licensee> <gvl>{0,1}</gvl> </ids></pre>
Source	<xsd:element name="ids" type="ids" />

Element contributor / www

Namespace	No namespace
-----------	--------------

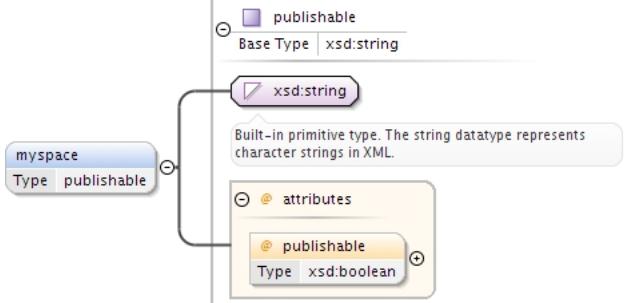
Diagram	<p>This Element is a container for the important web addresses and phone of the associated element (contributor e.g....)</p>						
Type	www						
Properties	<table border="1"> <tr> <td>content:</td><td>complex</td></tr> <tr> <td>minOccurs:</td><td>0</td></tr> <tr> <td>maxOccurs:</td><td>1</td></tr> </table>	content:	complex	minOccurs:	0	maxOccurs:	1
content:	complex						
minOccurs:	0						
maxOccurs:	1						
Model	ALL/facebook{0,1} myspace{0,1} homepage{0,1} twitter{0,1} phone{0,1})						
Children	facebook, homepage, myspace, phone, twitter						
Instance	<pre><www> <facebook publishable="">{0,1}</facebook> <myspace publishable="">{0,1}</myspace> <homepage publishable="">{0,1}</homepage> <twitter publishable="">{0,1}</twitter> <phone publishable="">{0,1}</phone> </www></pre>						
Source	<xsd:element name="www" type="www" maxOccurs="1" minOccurs="0" />						

Element www / facebook

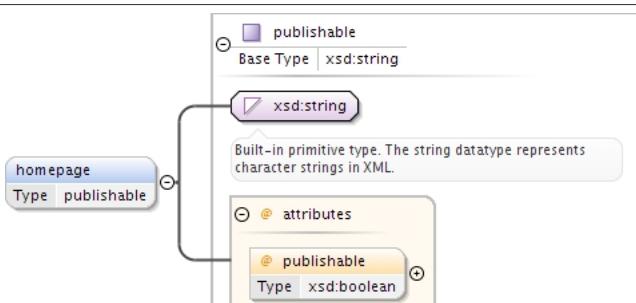
Namespace	No namespace										
Diagram	<p>Built-in primitive type. The string datatype represents character strings in XML.</p>										
Type	publishable										
Properties	<table border="1"> <tr> <td>content:</td><td>complex</td></tr> <tr> <td>minOccurs:</td><td>0</td></tr> <tr> <td>maxOccurs:</td><td>1</td></tr> </table>	content:	complex	minOccurs:	0	maxOccurs:	1				
content:	complex										
minOccurs:	0										
maxOccurs:	1										
Attributes	<table border="1"> <thead> <tr> <th>QName</th><th>Type</th><th>Fixed</th><th>Default</th><th>Use</th></tr> </thead> <tbody> <tr> <td>publishable</td><td>xsd:boolean</td><td></td><td></td><td>optional</td></tr> </tbody> </table>	QName	Type	Fixed	Default	Use	publishable	xsd:boolean			optional
QName	Type	Fixed	Default	Use							
publishable	xsd:boolean			optional							
Source	<xsd:element name="facebook" type="publishable" maxOccurs="1" minOccurs="0" />										

Element www / myspace

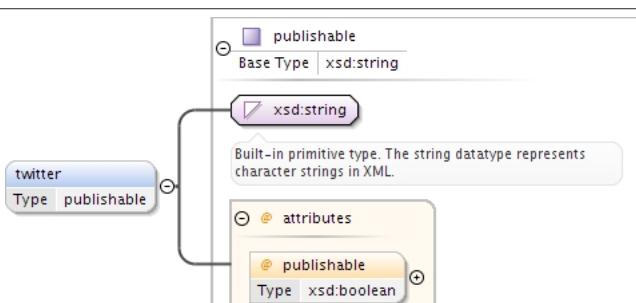
Namespace	No namespace
-----------	--------------

Diagram											
Type	publishable										
Properties	<table border="1"> <tr> <td>content:</td><td>complex</td></tr> <tr> <td>minOccurs:</td><td>0</td></tr> <tr> <td>maxOccurs:</td><td>1</td></tr> </table>	content:	complex	minOccurs:	0	maxOccurs:	1				
content:	complex										
minOccurs:	0										
maxOccurs:	1										
Attributes	<table border="1"> <thead> <tr> <th>QName</th><th>Type</th><th>Fixed</th><th>Default</th><th>Use</th></tr> </thead> <tbody> <tr> <td>publishable</td><td>xsd:boolean</td><td></td><td></td><td>optional</td></tr> </tbody> </table>	QName	Type	Fixed	Default	Use	publishable	xsd:boolean			optional
QName	Type	Fixed	Default	Use							
publishable	xsd:boolean			optional							
Source	<xsd:element name="myspace" type="publishable" maxOccurs="1" minOccurs="0" />										

Element www / homepage

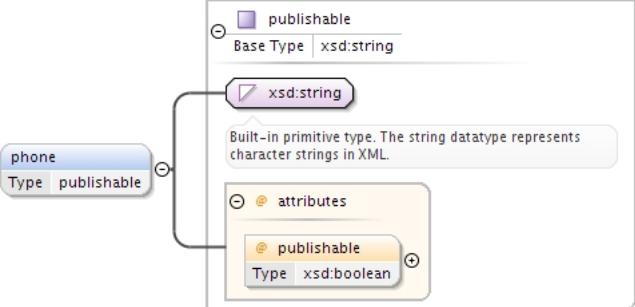
Namespace	No namespace										
Diagram											
Type	publishable										
Properties	<table border="1"> <tr> <td>content:</td><td>complex</td></tr> <tr> <td>minOccurs:</td><td>0</td></tr> <tr> <td>maxOccurs:</td><td>1</td></tr> </table>	content:	complex	minOccurs:	0	maxOccurs:	1				
content:	complex										
minOccurs:	0										
maxOccurs:	1										
Attributes	<table border="1"> <thead> <tr> <th>QName</th><th>Type</th><th>Fixed</th><th>Default</th><th>Use</th></tr> </thead> <tbody> <tr> <td>publishable</td><td>xsd:boolean</td><td></td><td></td><td>optional</td></tr> </tbody> </table>	QName	Type	Fixed	Default	Use	publishable	xsd:boolean			optional
QName	Type	Fixed	Default	Use							
publishable	xsd:boolean			optional							
Source	<xsd:element name="homepage" type="publishable" maxOccurs="1" minOccurs="0" />										

Element www / twitter

Namespace	No namespace
Diagram	
Type	publishable

Properties	content: complex minOccurs: 0 maxOccurs: 1				
Attributes	QName	Type	Fixed	Default	Use
	publishable	xsd:boolean			optional
Source	<xsd:element name="twitter" type="publishable" maxOccurs="1" minOccurs="0" />				

Element www / phone

Namespace	No namespace										
Diagram	 <pre> classDiagram class phone { <<publishable>> <<xsd:string>> <<attributes>> <<publishable>> <<xsd:boolean>> } phone < -- publishable publishable < -- xsd:string xsd:string < -- attributes attributes < -- publishable </pre>										
Type	publishable										
Properties	content: complex minOccurs: 0 maxOccurs: 1										
Attributes	<table border="1"> <tr> <th>QName</th><th>Type</th><th>Fixed</th><th>Default</th><th>Use</th></tr> <tr> <td>publishable</td><td>xsd:boolean</td><td></td><td></td><td>optional</td></tr> </table>	QName	Type	Fixed	Default	Use	publishable	xsd:boolean			optional
QName	Type	Fixed	Default	Use							
publishable	xsd:boolean			optional							
Source	<xsd:element name="phone" type="publishable" maxOccurs="1" minOccurs="0" />										

Element item / information

Namespace	No namespace
-----------	--------------

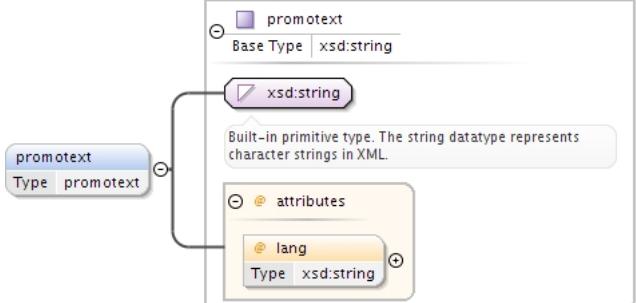
Diagram	
Type	information
Properties	content: complex
Model	ALL(texts{0,1} physical_release_datetime digital_release_datetime playlength{0,1} num{0,1} setnum{0,1} suggested_prelistening_offset{0,1} origin_country{0,1} main_language{0,1} related{0,1})
Children	digital_release_datetime, main_language, num, origin_country, physical_release_datetime, playlength, related, setnum, suggested_prelistening_offset, texts
Instance	<pre><information> <texts>{0,1}</texts> <physical_release_datetime>{1,1}</physical_release_datetime> <digital_release_datetime>{1,1}</digital_release_datetime> <playlength>{0,1}</playlength> <num>{0,1}</num> <setnum>{0,1}</setnum> <suggested_prelistening_offset>{0,1}</suggested_prelistening_offset> <origin_country>{0,1}</origin_country> <main_language>{0,1}</main_language> <related>{0,1}</related> </information></pre>
Source	<code><xsd:element name="information" type="information" /></code>

Element information / texts

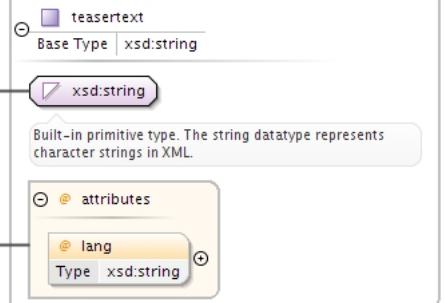
Namespace	No namespace
Diagram	
Type	texts
Properties	content: complex

	minOccurs: 0 maxOccurs: 1
Model	promotext*, teasertext*
Children	promotext, teasertext
Instance	<pre><texts> <promotext lang="">{0,unbounded}</promotext> <teasertext lang="">{0,unbounded}</teasertext> </texts></pre>
Source	<code><xsd:element name="texts" type="text" maxOccurs="1" minOccurs="0" /></code>

Element texts / promotext

Namespace	No namespace										
Diagram											
Type	promotext										
Properties	content: complex minOccurs: 0 maxOccurs: unbounded										
Attributes	<table border="1"> <thead> <tr> <th>QName</th> <th>Type</th> <th>Fixed</th> <th>Default</th> <th>Use</th> </tr> </thead> <tbody> <tr> <td>lang</td> <td>xsd:string</td> <td></td> <td></td> <td>optional</td> </tr> </tbody> </table>	QName	Type	Fixed	Default	Use	lang	xsd:string			optional
QName	Type	Fixed	Default	Use							
lang	xsd:string			optional							
Source	<code><xsd:element name="promotext" type="promotext" maxOccurs="unbounded" minOccurs="0" /></code>										

Element texts / teasertext

Namespace	No namespace										
Diagram											
Type	teasertext										
Properties	content: complex minOccurs: 0 maxOccurs: unbounded										
Attributes	<table border="1"> <thead> <tr> <th>QName</th> <th>Type</th> <th>Fixed</th> <th>Default</th> <th>Use</th> </tr> </thead> <tbody> <tr> <td>lang</td> <td>xsd:string</td> <td></td> <td></td> <td>optional</td> </tr> </tbody> </table>	QName	Type	Fixed	Default	Use	lang	xsd:string			optional
QName	Type	Fixed	Default	Use							
lang	xsd:string			optional							
Source	<code><xsd:element name="teasertext" type="teasertext" maxOccurs="unbounded" minOccurs="0" /></code>										

Element information / physical_release_datetime

Namespace	No namespace
-----------	--------------

Diagram	
Type	datetimeGMT
Properties	content: simple
Facets	pattern $\d{4}-\d{2}-\d{2}$ $\d{2}:\d{2}:\d{2}\ \text{GMT}$ $+\d{2}:\d{2}$
Source	<xsd:element name="physical_release_datetime" type="datetimeGMT" />

Element information / digital_release_datetime

Namespace	No namespace
Diagram	
Type	datetimeGMT
Properties	content: simple
Facets	pattern $\d{4}-\d{2}-\d{2}$ $\d{2}:\d{2}:\d{2}\ \text{GMT}$ $+\d{2}:\d{2}$
Source	<xsd:element name="digital_release_datetime" type="datetimeGMT" />

Element information / playlength

Namespace	No namespace
Diagram	 Built-in derived type. The integer datatype is derived from decimal by fixing the value of fractionDigits to be 0. This...
Type	xsd:integer
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="playlength" type="xsd:integer" maxOccurs="1" minOccurs="0" />

Element information / num

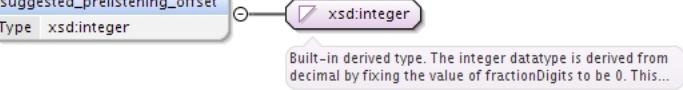
Namespace	No namespace
Diagram	 Built-in derived type. The integer datatype is derived from decimal by fixing the value of fractionDigits to be 0. This...
Type	xsd:integer
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="num" type="xsd:integer" maxOccurs="1" minOccurs="0" />

Element information / setnum

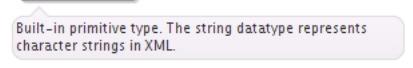
Namespace	No namespace
Diagram	 Built-in derived type. The integer datatype is derived from decimal by fixing the value of fractionDigits to be 0. This...

Type	xsd:integer
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="setnum" type="xsd:integer" maxOccurs="1" minOccurs="0" />

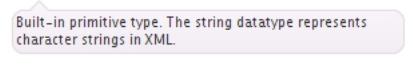
Element information / suggested_prelistening_offset

Namespace	No namespace
Diagram	
Type	xsd:integer
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="suggested_prelistening_offset" type="xsd:integer" maxOccurs="1" minOccurs="0" />

Element information / origin_country

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="origin_country" type="xsd:string" maxOccurs="1" minOccurs="0" />

Element information / main_language

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="main_language" type="xsd:string" maxOccurs="1" minOccurs="0" />

Element information / related

Namespace	No namespace
-----------	--------------

Diagram	
Type	related
Properties	<p>content: complex</p> <p>minOccurs: 0</p> <p>maxOccurs: 1</p>
Model	physical_distributor* , utube{0,1} , bundle*
Children	bundle, physical_distributor, utube
Instance	<pre><related> <physical_distributor publishable="">{0,unbounded}</physical_distributor> <utube>{0,1}</utube> <bundle>{0,unbounded}</bundle> </related></pre>
Source	<code><xsd:element name="related" type="related" maxOccurs="1" minOccurs="0" /></code>

Element related / physical_distributor

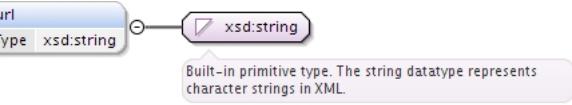
Namespace	No namespace										
Diagram											
Type	physical_distributor										
Properties	<p>content: complex</p> <p>minOccurs: 0</p> <p>maxOccurs: unbounded</p>										
Attributes	<table border="1"> <thead> <tr> <th>QName</th> <th>Type</th> <th>Fixed</th> <th>Default</th> <th>Use</th> </tr> </thead> <tbody> <tr> <td>publishable</td> <td>xsd:boolean</td> <td></td> <td></td> <td>optional</td> </tr> </tbody> </table>	QName	Type	Fixed	Default	Use	publishable	xsd:boolean			optional
QName	Type	Fixed	Default	Use							
publishable	xsd:boolean			optional							
Source	<code><xsd:element name="physical_distributor" type="physical_distributor" maxOccurs="unbounded" minOccurs="0" /></code>										

Element related / utube

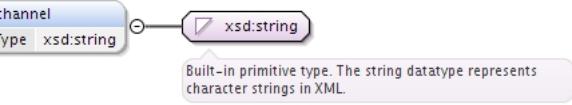
Namespace	No namespace
Diagram	

Type	utube
Properties	content: complex minOccurs: 0 maxOccurs: 1
Model	ALL(url{0,1} channel{0,1})
Children	channel, url
Instance	<pre><utube> <url>{0,1}</url> <channel>{0,1}</channel> </utube></pre>
Source	<code><xsd:element name="utube" type="utube" maxOccurs="1" minOccurs="0" /></code>

Element utube / url

Namespace	No namespace
Diagram	 <p>The diagram shows a rounded rectangle labeled "url" with a small circle icon to its right, indicating it's a complex type. An arrow points from "url" to a purple hexagon icon representing "xsd:string". A tooltip below the arrow states: "Built-in primitive type. The string datatype represents character strings in XML."</p>
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<code><xsd:element name="url" type="xsd:string" maxOccurs="1" minOccurs="0" /></code>

Element utube / channel

Namespace	No namespace
Diagram	 <p>The diagram shows a rounded rectangle labeled "channel" with a small circle icon to its right, indicating it's a complex type. An arrow points from "channel" to a purple hexagon icon representing "xsd:string". A tooltip below the arrow states: "Built-in primitive type. The string datatype represents character strings in XML."</p>
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<code><xsd:element name="channel" type="xsd:string" maxOccurs="1" minOccurs="0" /></code>

Element related / bundle

Namespace	No namespace
-----------	--------------

Diagram	<p>On bundle level, there are information on how to handle a collection of "items". This is mainly an album/ep/single. A...</p>						
Type	bundle						
Properties	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">content:</td><td style="padding: 2px;">complex</td></tr> <tr> <td style="padding: 2px;">minOccurs:</td><td style="padding: 2px;">0</td></tr> <tr> <td style="padding: 2px;">maxOccurs:</td><td style="padding: 2px;">unbounded</td></tr> </table>	content:	complex	minOccurs:	0	maxOccurs:	unbounded
content:	complex						
minOccurs:	0						
maxOccurs:	unbounded						
Model	ALL(displayname{0,1} name{0,1} version{0,1} display_artistname{0,1} ids items{0,1} contributors{0,1} information{0,1} license_basis{0,1} license_specifics{0,1} reporting{0,1} tags{0,1} files{0,1} purchase{0,1})						
Children	contributors, display_artistname, displayname, files, ids, information, items, license_basis, license_specifics, name, purchase, reporting, tags, version						
Instance	<pre style="font-family: monospace; padding: 10px;"> <bundle> <displayname>{0,1}</displayname> <name>{0,1}</name> <version>{0,1}</version> <display_artistname>{0,1}</display_artistname> <ids>{1,1}</ids> <items>{0,1}</items> <contributors>{0,1}</contributors> <information>{0,1}</information> <license_basis>{0,1}</license_basis> <license_specifics>{0,1}</license_specifics> <reporting>{0,1}</reporting> <tags>{0,1}</tags> <files>{0,1}</files> <purchase>{0,1}</purchase> </bundle></pre>						
Source	<xsd:element name="bundle" type="bundle" maxOccurs="unbounded" minOccurs="0" />						

Element bundle / contributors

Namespace	No namespace						
Diagram	<pre> classDiagram class contributors { <<Type: contributors>> } class contributor { <<Type: contributor>> } contributors "1..>> contributor </pre> <p>This element contains a list of contributor.</p>						
Type	contributors						
Properties	<table border="1"> <tr> <td>content:</td> <td>complex</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	complex	minOccurs:	0	maxOccurs:	1
content:	complex						
minOccurs:	0						
maxOccurs:	1						
Model	contributor+						
Children	contributor						
Instance	<pre> <contributors> <contributor num="">{1,unbounded}</contributor> </contributors> </pre>						
Source	<pre> <xsd:element name="contributors" type="contributors" maxOccurs="1" minOccurs="0" /> </pre>						

Element bundle / information

Namespace	No namespace						
Diagram	<pre> classDiagram class information { <<Type: information>> } class texts { <<Type: texts>> } class physical_release_datetime { <<Type: datetimeGMT>> } class digital_release_datetime { <<Type: datetimeGMT>> } class playlength { <<Type: xsd:integer>> } class num { <<Type: xsd:integer>> } class setnum { <<Type: xsd:integer>> } class suggested_prelistening_offset { <<Type: xsd:integer>> } class origin_country { <<Type: xsd:string>> } class main_language { <<Type: xsd:string>> } class related { <<Type: related>> } information "0..1" --> texts information "0..1" --> physical_release_datetime information "0..1" --> digital_release_datetime information "0..1" --> playlength information "0..1" --> num information "0..1" --> setnum information "0..1" --> suggested_prelistening_offset information "0..1" --> origin_country information "0..1" --> main_language information "0..1" --> related </pre> <p>This element contains important data for an item/file. Multilingual promotexts and teasertexts, dates of physical and...</p>						
Type	information						
Properties	<table border="1"> <tr> <td>content:</td> <td>complex</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	complex	minOccurs:	0	maxOccurs:	1
content:	complex						
minOccurs:	0						
maxOccurs:	1						
Model	ALL(texts{0,1} physical_release_datetime digital_release_datetime playlength{0,1} num{0,1} setnum{0,1} suggested_prelistening_offset{0,1} origin_country{0,1} main_language{0,1} related{0,1})						
Children	digital_release_datetime, main_language, num, origin_country, physical_release_datetime, playlength, related, setnum, suggested_prelistening_offset, texts						

Instance	<pre><information> <texts>{0,1}</texts> <physical_release_datetime>{1,1}</physical_release_datetime> <digital_release_datetime>{1,1}</digital_release_datetime> <playlength>{0,1}</playlength> <num>{0,1}</num> <setnum>{0,1}</setnum> <suggested_prelistening_offset>{0,1}</suggested_prelistening_offset> <origin_country>{0,1}</origin_country> <main_language>{0,1}</main_language> <related>{0,1}</related> </information></pre>
Source	<pre><xsd:element name="information" type="information" maxOccurs="1" minOccurs="0"/></pre>

Element bundle / license_basis

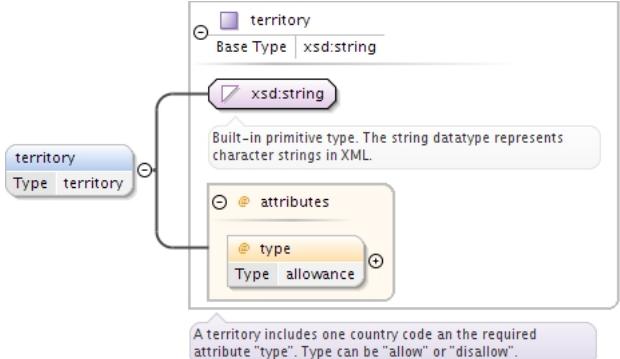
Namespace	No namespace						
Diagram	<pre> classDiagram class license_basis { territorial timeframe pricing streaming_allowed channels as_on_bundle } license_basis < -- license_basis </pre> <p>This element includes the basic rules and information under which this bundle is provided.</p>						
Type	license_basis						
Properties	<table border="1"> <tr> <td>content:</td> <td>complex</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	complex	minOccurs:	0	maxOccurs:	1
content:	complex						
minOccurs:	0						
maxOccurs:	1						
Model	ALL(territorial{0,1} timeframe{0,1} pricing{0,1} streaming_allowed{0,1} channels{0,1} as_on_bundle{0,1})						
Children	as_on_bundle, channels, pricing, streaming_allowed, territorial, timeframe						
Instance	<pre><license_basis> <territorial>{0,1}</territorial> <timeframe>{0,1}</timeframe> <pricing>{0,1}</pricing> <streaming_allowed>{0,1}</streaming_allowed> <channels>{0,1}</channels> <as_on_bundle>{0,1}</as_on_bundle> </license_basis></pre>						
Source	<pre><xsd:element name="license_basis" type="license_basis" maxOccurs="1" minOccurs="0"/></pre>						

Element license_basis / territorial

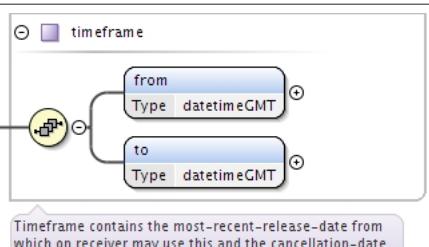
Namespace	No namespace		
Diagram	<pre> classDiagram class territorial { <<many-to-many>> territory } territorial < -- territorial </pre> <p>This Element is a container for territories. There should be a entry for all territories with a attribute if...</p>		
Type	territorial		
Properties	<table border="1"> <tr> <td>content:</td> <td>complex</td> </tr> </table>	content:	complex
content:	complex		

	minOccurs: 0
	maxOccurs: 1
Model	territory*
Children	territory
Instance	<territorial> <territory type="">{0,unbounded}</territory> </territorial>
Source	<xsd:element name="territorial" type="territorial" maxOccurs="1" minOccurs="0"/>

Element territorial / territory

Namespace	No namespace										
Diagram											
Type	territory										
Properties	<p>content: complex</p> <p>minOccurs: 0</p> <p>maxOccurs: unbounded</p>										
Attributes	<table border="1"> <thead> <tr> <th>QName</th> <th>Type</th> <th>Fixed</th> <th>Default</th> <th>Use</th> </tr> </thead> <tbody> <tr> <td>type</td> <td>allowance</td> <td></td> <td></td> <td>optional</td> </tr> </tbody> </table>	QName	Type	Fixed	Default	Use	type	allowance			optional
QName	Type	Fixed	Default	Use							
type	allowance			optional							
Source	<xsd:element name="territory" type="territory" maxOccurs="unbounded" minOccurs="0"/>										

Element license_basis / timeframe

Namespace	No namespace
Diagram	
Type	timeframe
Properties	<p>content: complex</p> <p>minOccurs: 0</p> <p>maxOccurs: 1</p>
Model	from , to
Children	from, to
Instance	<timeframe> <from>{1,1}</from> <to>{1,1}</to> </timeframe>
Source	<xsd:element name="timeframe" type="timeframe" maxOccurs="1" minOccurs="0"/>

Element timeframe / from

Namespace	No namespace
Diagram	<pre> graph LR from[from] --> datetimeGMT[datetimeGMT] </pre>
Type	datetimeGMT
Properties	content: simple
Facets	pattern <pre>\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2} GMT\ +\d{2}:\d{2}</pre>
Source	<xsd:element name="from" type="datetimeGMT" />

Element timeframe / to

Namespace	No namespace
Diagram	<pre> graph LR to[to] --> datetimeGMT[datetimeGMT] </pre>
Type	datetimeGMT
Properties	content: simple
Facets	pattern <pre>\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2} GMT\ +\d{2}:\d{2}</pre>
Source	<xsd:element name="to" type="datetimeGMT" />

Element license_basis / pricing

Namespace	No namespace
Diagram	<pre> graph LR pricing(pricing) --> pricecode(pricecode) pricing --> wholesale(wholesale) </pre> <p>Pricecode is an arbitrary-info. An explicitly given wholesale-price overrides the basic pricecode-given-wp. Most...</p>
Type	pricing
Properties	content: complex minOccurs: 0 maxOccurs: 1
Model	ALL(pricecode{0,1} wholesale{0,1})
Children	pricecode, wholesale
Instance	<pricing> <pricecode>{0,1}</pricecode> <wholesale>{0,1}</wholesale> </pricing>
Source	<xsd:element name="pricing" type="pricing" maxOccurs="1" minOccurs="0" />

Element pricing / pricecode

Namespace	No namespace
Diagram	<pre> graph LR pricecode[pricecode] --> xsdString[xsd:string] </pre> <p>Built-in primitive type. The string datatype represents character strings in XML.</p>

Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="pricecode" type="xsd:string" maxOccurs="1" minOccurs="0" />

Element pricing / wholesale

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="wholesale" type="xsd:string" maxOccurs="1" minOccurs="0" />

Element license_basis / streaming_allowed

Namespace	No namespace
Diagram	
Type	xsd:boolean
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="streaming_allowed" type="xsd:boolean" maxOccurs="1" minOccurs="0" />

Element license_basis / channels

Namespace	No namespace
Diagram	
Type	channels
Properties	content: complex minOccurs: 0 maxOccurs: 1
Model	channel*
Children	channel
Instance	<channels> <channel type="">{0,unbounded}</channel> </channels>
Source	<xsd:element name="channels" type="channels" maxOccurs="1" minOccurs="0" />

Element channels / channel

Namespace	No namespace
-----------	--------------

Diagram											
Type	channel										
Properties	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">content:</td><td style="padding: 2px;">complex</td></tr> <tr> <td style="padding: 2px;">minOccurs:</td><td style="padding: 2px;">0</td></tr> <tr> <td style="padding: 2px;">maxOccurs:</td><td style="padding: 2px;">unbounded</td></tr> </table>	content:	complex	minOccurs:	0	maxOccurs:	unbounded				
content:	complex										
minOccurs:	0										
maxOccurs:	unbounded										
Attributes	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">QName</th><th style="text-align: left; padding: 2px;">Type</th><th style="text-align: left; padding: 2px;">Fixed</th><th style="text-align: left; padding: 2px;">Default</th><th style="text-align: left; padding: 2px;">Use</th></tr> </thead> <tbody> <tr> <td style="padding: 2px;">type</td><td style="padding: 2px;">allowance</td><td style="padding: 2px;"></td><td style="padding: 2px;"></td><td style="padding: 2px;">required</td></tr> </tbody> </table>	QName	Type	Fixed	Default	Use	type	allowance			required
QName	Type	Fixed	Default	Use							
type	allowance			required							
Source	<code><xsd:element name="channel" type="channel" maxOccurs="unbounded" minOccurs="0" /></code>										

Element license_basis / as_on_bundle

Namespace	No namespace						
Diagram							
Type	xsd:boolean						
Properties	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">content:</td><td style="padding: 2px;">simple</td></tr> <tr> <td style="padding: 2px;">minOccurs:</td><td style="padding: 2px;">0</td></tr> <tr> <td style="padding: 2px;">maxOccurs:</td><td style="padding: 2px;">1</td></tr> </table>	content:	simple	minOccurs:	0	maxOccurs:	1
content:	simple						
minOccurs:	0						
maxOccurs:	1						
Source	<code><xsd:element name="as_on_bundle" type="xsd:boolean" maxOccurs="1" minOccurs="0" /></code>						

Element bundle / license_specifics

Namespace	No namespace						
Diagram							
Type	license_specifics						
Properties	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">content:</td><td style="padding: 2px;">complex</td></tr> <tr> <td style="padding: 2px;">minOccurs:</td><td style="padding: 2px;">0</td></tr> <tr> <td style="padding: 2px;">maxOccurs:</td><td style="padding: 2px;">1</td></tr> </table>	content:	complex	minOccurs:	0	maxOccurs:	1
content:	complex						
minOccurs:	0						
maxOccurs:	1						
Model	rules						
Children	rules						
Instance	<code><license_specifics> <rules>{1,1}</rules> </license_specifics></code>						
Source	<code><xsd:element name="license_specifics" type="license_specifics" maxOccurs="1" minOccurs="0" /></code>						

Element license_specifics / rules

Namespace	No namespace
Diagram	<p>This element is a container for rules. It needs an ordered mode here - first come first match.</p>
Type	rules
Properties	content: complex
Model	rule*
Children	rule
Instance	<pre><rules> <rule num="">{0,unbounded}</rule> </rules></pre>
Source	<code><xsd:element name="rules" type="rules"/></code>

Element rules / rule

Namespace	No namespace										
Diagram	<p>A rule must include a "if"-element and a "then"-element to shape a legal instruction. It can also include a...</p>										
Type	rule										
Properties	content: complex minOccurs: 0 maxOccurs: unbounded										
Model	if , then , else{0,1}										
Children	else, if, then										
Instance	<pre><rule num=""> <if>{1,1}</if> <then>{1,1}</then> <else>{0,1}</else> </rule></pre>										
Attributes	<table border="1"> <thead> <tr> <th>QName</th> <th>Type</th> <th>Fixed</th> <th>Default</th> <th>Use</th> </tr> </thead> <tbody> <tr> <td>num</td> <td>xsd:integer</td> <td></td> <td></td> <td>optional</td> </tr> </tbody> </table>	QName	Type	Fixed	Default	Use	num	xsd:integer			optional
QName	Type	Fixed	Default	Use							
num	xsd:integer			optional							
Source	<code><xsd:element name="rule" type="rule" maxOccurs="unbounded" minOccurs="0"/></code>										

Element rule / if

Namespace	No namespace
-----------	--------------

Diagram	
Type	if
Properties	content: complex
Model	what , operator , value
Children	operator, value, what
Instance	<pre><if> <what>{1,1}</what> <operator>{1,1}</operator> <value>{1,1}</value> </if></pre>
Source	<code><xsd:element name="if" type="if"/></code>

Element if / what

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple
Source	<code><xsd:element name="what" type="xsd:string"/></code>

Element if / operator

Namespace	No namespace
Diagram	
Type	operator
Properties	content: simple
Facets	enumeration equals enumeration before enumeration after enumeration contains enumeration containedin
Source	<code><xsd:element name="operator" type="operator"/></code>

Element if / value

Namespace	No namespace
Diagram	
Type	xsd:string

Properties	content: simple
Source	<xsd:element name="value" type="xsd:string" />

Element rule / then

Namespace	No namespace
Diagram	<p>This element must be the second in a rule and includes information "echo" for debugging output and can include an...</p>
Type	then
Properties	content: complex
Model	echo{0,1} , break{0,1}
Children	break, echo
Instance	<pre><then> <echo>{0,1}</echo> <break>{0,1}</break> </then></pre>
Source	<xsd:element name="then" type="then" />

Element then / echo

Namespace	No namespace
Diagram	<p>Built-in primitive type. The string datatype represents character strings in XML.</p>
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="echo" type="xsd:string" maxOccurs="1" minOccurs="0" />

Element then / break

Namespace	No namespace
Diagram	
Properties	minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="break" maxOccurs="1" minOccurs="0" />

Element rule / else

Namespace	No namespace
Diagram	<p>This element is optional. It includes information "proclaim" and can include an element "break" which means to not...</p>

Type	else
Properties	<p>content: complex</p> <p>minOccurs: 0</p> <p>maxOccurs: 1</p>
Model	proclaim*, break{0,1}
Children	break, proclaim
Instance	<pre><else> <proclaim>{0,unbounded}</proclaim> <break>{0,1}</break> </else></pre>
Source	<code><xsd:element name="else" type="else" maxOccurs="1" minOccurs="0" /></code>

Element else / proclaim

Namespace	No namespace
Diagram	
Type	proclaim
Properties	<p>content: complex</p> <p>minOccurs: 0</p> <p>maxOccurs: unbounded</p>
Model	what , for
Children	for, what
Instance	<pre><proclaim> <what>{1,1}</what> <for>{1,1}</for> </proclaim></pre>
Source	<code><xsd:element name="proclaim" type="proclaim" maxOccurs="unbounded" minOccurs="0" /></code>

Element proclaim / what

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple
Source	<code><xsd:element name="what" type="xsd:string" /></code>

Element proclaim / for

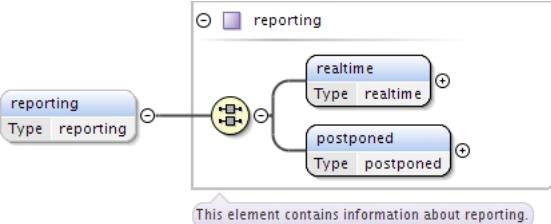
Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple

Source	<code><xsd:element name="for" type="xsd:string"/></code>
--------	--

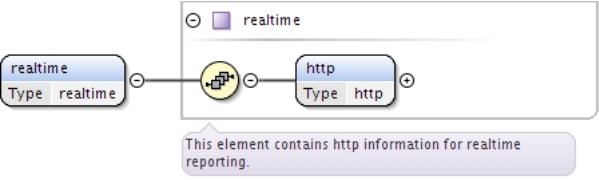
Element else / break

Namespace	No namespace
Diagram	
Properties	minOccurs: 0 maxOccurs: 1
Source	<code><xsd:element name="break" maxOccurs="1" minOccurs="0"/></code>

Element bundle / reporting

Namespace	No namespace
Diagram	 <p>This element contains information about reporting.</p>
Type	reporting
Properties	content: complex minOccurs: 0 maxOccurs: 1
Model	ALL(realtime postponed)
Children	postponed, realtime
Instance	<code><reporting> <realtime>{1,1}</realtime> <postponed>{1,1}</postponed> </reporting></code>
Source	<code><xsd:element name="reporting" type="reporting" maxOccurs="1" minOccurs="0"/></code>

Element reporting / realtime

Namespace	No namespace
Diagram	 <p>This element contains http information for realtime reporting.</p>
Type	realtime
Properties	content: complex
Model	http
Children	http
Instance	<code><realtime> <http>{1,1}</http> </realtime></code>
Source	<code><xsd:element name="realtime" type="realtime" /></code>

Element realtime / http

Namespace	No namespace
-----------	--------------

Diagram	<pre> classDiagram class http { <<Base Type action>> <<action (extension base)>> http Type http } class url { <<Type url>> url Type url } class type { <<Type httpmethods>> type Type httpmethods } class addheader { <<Type http_addheader>> addheader Type http_addheader } class addparams { <<Type http_addparams>> addparams Type http_addparams } http < -- url http < -- type http < -- addheader http < -- addparams </pre> <p>This element contains information about http-event.</p>
Type	http
Type hierarchy	<ul style="list-style-type: none"> • action • http
Properties	content: complex
Model	ALL(url type addheader addparams)
Children	addheader, addparams, type, url
Instance	<pre> <http> <url>{1,1}</url> <type>{1,1}</type> <addheader>{1,1}</addheader> <addparams>{1,1}</addparams> </http> </pre>
Source	<xsd:element name="http" type="http"/>

Element reporting / postponed

Namespace	No namespace
Diagram	<pre> classDiagram class postponed { <<postponed>> postponed Type postponed } class id { <<Type xsd:string>> id Type xsd:string } postponed < -- id </pre> <p>This element contains some info on reporting when doing the "usual" time-gap-reporting.</p>
Type	postponed
Properties	content: complex
Model	id
Children	id
Instance	<pre> <postponed> <id>{1,1}</id> </postponed> </pre>
Source	<xsd:element name="postponed" type="postponed"/>

Element postponed / id

Namespace	No namespace
Diagram	<pre> classDiagram class id { <<Type xsd:string>> id Type xsd:string } xsd:string </pre> <p>Built-in primitive type. The string datatype represents character strings in XML.</p>
Type	xsd:string
Properties	content: simple
Source	<xsd:element name="id" type="xsd:string"/>

Element bundle / tags

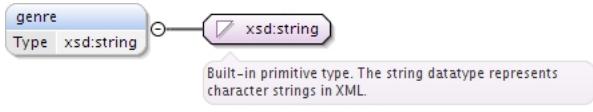
Namespace	No namespace
Diagram	<pre> classDiagram class tags { <<tags>> <<Type tags>> <<0..1>> <<genres>> <<bundle_only>> <<explicit_lyrics>> <<live>> <<accoustic>> <<instrumental>> } note over tags: This element contains information about genres and more. </pre>
Type	tags
Properties	<p>content: complex</p> <p>minOccurs: 0</p> <p>maxOccurs: 1</p>
Model	ALL(genres{0,1} bundle_only{0,1} explicit_lyrics{0,1} live{0,1} accoustic{0,1} instrumental{0,1})
Children	accoustic, bundle_only, explicit_lyrics, genres, instrumental, live
Instance	<pre> <tags> <genres>{0,1}</genres> <bundle_only>{0,1}</bundle_only> <explicit_lyrics>{0,1}</explicit_lyrics> <live>{0,1}</live> <accoustic>{0,1}</accoustic> <instrumental>{0,1}</instrumental> </tags> </pre>
Source	<xsd:element name="tags" type="tags" maxOccurs="1" minOccurs="0"/>

Element tags / genres

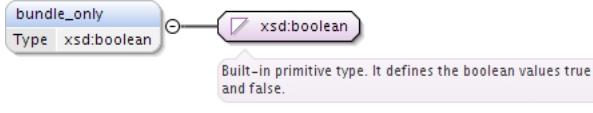
Namespace	No namespace
Diagram	<pre> classDiagram class genres { <<genres>> <<Type genres>> <<0..infinity>> <<genre>> <<Type xsd:string>> } note over genres: This element contains a list of genres. </pre>
Type	genres
Properties	<p>content: complex</p> <p>minOccurs: 0</p> <p>maxOccurs: 1</p>
Model	genre*
Children	genre
Instance	<pre> <genres> <genre>{0..unbounded}</genre> </genres> </pre>
Source	<xsd:element name="genres" type="genres" maxOccurs="1" minOccurs="0"/>

Element genres / genre

Namespace	No namespace
-----------	--------------

Diagram	 A diagram showing the 'genre' element mapped to the 'xsd:string' type. A callout box indicates that 'xsd:string' is a 'Built-in primitive type. The string datatype represents character strings in XML.'
Type	xsd:string
Properties	<p>content: simple</p> <p>minOccurs: 0</p> <p>maxOccurs: unbounded</p>
Source	<xsd:element name="genre" type="xsd:string" maxOccurs="unbounded" minOccurs="0" />

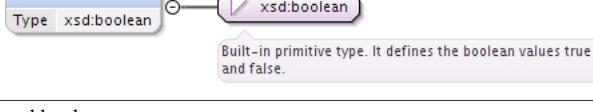
Element tags / bundle_only

Namespace	No namespace
Diagram	 A diagram showing the 'bundle_only' element mapped to the 'xsd:boolean' type. A callout box indicates that 'xsd:boolean' is a 'Built-in primitive type. It defines the boolean values true and false.'
Type	xsd:boolean
Properties	<p>content: simple</p> <p>minOccurs: 0</p> <p>maxOccurs: 1</p>
Source	<xsd:element name="bundle_only" type="xsd:boolean" maxOccurs="1" minOccurs="0" />

Element tags / explicit_lyrics

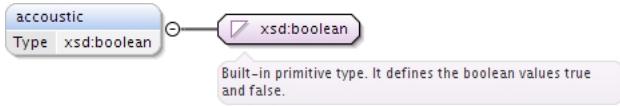
Namespace	No namespace
Diagram	 A diagram showing the 'explicit_lyrics' element mapped to the 'explicitLyrics' type.
Type	explicitLyrics
Properties	<p>content: simple</p> <p>minOccurs: 0</p> <p>maxOccurs: 1</p>
Facets	<p>enumeration true</p> <p>enumeration false</p> <p>enumeration cleaned</p>
Source	<xsd:element name="explicit_lyrics" type="explicitLyrics" maxOccurs="1" minOccurs="0" />

Element tags / live

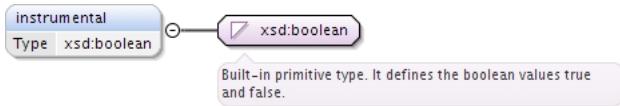
Namespace	No namespace
Diagram	 A diagram showing the 'live' element mapped to the 'xsd:boolean' type. A callout box indicates that 'xsd:boolean' is a 'Built-in primitive type. It defines the boolean values true and false.'
Type	xsd:boolean
Properties	<p>content: simple</p> <p>minOccurs: 0</p> <p>maxOccurs: 1</p>
Source	<xsd:element name="live" type="xsd:boolean" maxOccurs="1" minOccurs="0" />

Element tags / acoustic

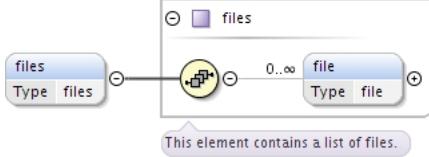
Namespace	No namespace
-----------	--------------

Diagram	
Type	xsd:boolean
Properties	<p>content: simple</p> <p>minOccurs: 0</p> <p>maxOccurs: 1</p>
Source	<xsd:element name="acoustic" type="xsd:boolean" maxOccurs="1" minOccurs="0" />

Element tags / instrumental

Namespace	No namespace
Diagram	
Type	xsd:boolean
Properties	<p>content: simple</p> <p>minOccurs: 0</p> <p>maxOccurs: 1</p>
Source	<xsd:element name="instrumental" type="xsd:boolean" maxOccurs="1" minOccurs="0" />

Element bundle / files

Namespace	No namespace
Diagram	
Type	files
Properties	<p>content: complex</p> <p>minOccurs: 0</p> <p>maxOccurs: 1</p>
Model	file*
Children	file
Instance	<pre><files> <file>{0..unbounded}</file> </files></pre>
Source	<xsd:element name="files" type="files" maxOccurs="1" minOccurs="0" />

Element files / file

Namespace	No namespace
-----------	--------------

Diagram	<p>This element contains information and location of a file.</p>
Type	file
Properties	content: complex minOccurs: 0 maxOccurs: unbounded
Model	ALL(location type{0,1} filetype{0,1} samplerate{0,1} samplesize{0,1} bitrate{0,1} bitratatype{0,1} codec{0,1} codecsettings{0,1} bytes{0,1} checksums channels{0,1} dimension{0,1} decryptinfo{0,1})
Children	bitrate, bitratatype, bytes, channels, checksums, codec, codecsettings, decryptinfo, dimension, filetype, location, samplerate, samplesize, type
Instance	<pre><file> <location>{1,1}</location> <type>{0,1}</type> <filetype>{0,1}</filetype> <samplerate>{0,1}</samplerate> <samplesize>{0,1}</samplesize> <bitrate>{0,1}</bitrate> <bitratatype>{0,1}</bitratatype> <codec>{0,1}</codec> <codecsettings>{0,1}</codecsettings> <bytes>{0,1}</bytes> <checksums>{1,1}</checksums> <channels>{0,1}</channels> <dimension>{0,1}</dimension> <decryptinfo>{0,1}</decryptinfo> </file></pre>
Source	<code><xsd:element name="file" type="file" maxOccurs="unbounded" minOccurs="0" /></code>

Element file / location

Namespace	No namespace
Diagram	<pre> classDiagram class fileLocation { path : xsd:string http : fileHttp ftp : fileFtp } location <--> fileLocation note over fileLocation: This element contains the path to the corresponding file. </pre>
Type	fileLocation
Properties	content: complex
Model	ALL(path{0,1} http{0,1} ftp{0,1})
Children	ftp, http, path
Instance	<pre> <location> <path>{0,1}</path> <http>{0,1}</http> <ftp>{0,1}</ftp> </location> </pre>
Source	<xsd:element name="location" type="fileLocation"/>

Element fileLocation / path

Namespace	No namespace						
Diagram	<pre> classDiagram class xsd:string path <--> xsd:string note over xsd:string: Built-in primitive type. The string datatype represents character strings in XML. </pre>						
Type	xsd:string						
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	simple	minOccurs:	0	maxOccurs:	1
content:	simple						
minOccurs:	0						
maxOccurs:	1						
Source	<xsd:element name="path" type="xsd:string" maxOccurs="1" minOccurs="0" />						

Element fileLocation / http

Namespace	No namespace
Diagram	<pre> classDiagram class fileHttp { url : url user : xsd:string pass : xsd:string expiresdatetime : datetimeGMT } fileHttp < -- action http <--> fileHttp note over fileHttp: This element contains information about http access to file. </pre>
Type	fileHttp
Type hierarchy	• action

	<ul style="list-style-type: none"> • fileHttp
Properties	content: complex minOccurs: 0 maxOccurs: 1
Model	ALL(url user{0,1} pass{0,1} expiresdatetime)
Children	expiresdatetime, pass, url, user
Instance	<pre><http> <url>{1,1}</url> <user>{0,1}</user> <pass>{0,1}</pass> <expiresdatetime>{1,1}</expiresdatetime> </http></pre>
Source	<code><xsd:element name="http" type="fileHttp" maxOccurs="1" minOccurs="0"/></code>

Element fileHttp / url

Namespace	No namespace
Diagram	<pre> graph LR url[url] --> url </pre>
Type	url
Properties	content: simple
Source	<code><xsd:element name="url" type="url" /></code>

Element fileHttp / user

Namespace	No namespace
Diagram	<pre> graph LR user[user] --> xsdString[xsd:string] </pre> <p>Built-in primitive type. The string datatype represents character strings in XML.</p>
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<code><xsd:element name="user" type="xsd:string" maxOccurs="1" minOccurs="0"/></code>

Element fileHttp / pass

Namespace	No namespace
Diagram	<pre> graph LR pass[pass] --> xsdString[xsd:string] </pre> <p>Built-in primitive type. The string datatype represents character strings in XML.</p>
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<code><xsd:element name="pass" type="xsd:string" maxOccurs="1" minOccurs="0"/></code>

Element fileHttp / expiresdatetime

Namespace	No namespace
Diagram	<pre> graph LR expiresDatetime[expiresdatetime] --> datetimeGMT[datetimeGMT] </pre>

Type	datetimeGMT
Properties	content: simple
Facets	pattern \d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2} GMT\ +\d{2}:\d{2}
Source	<xsd:element name="expiresdatetime" type="datetimeGMT" />

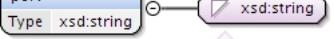
Element fileLocation / ftp

Namespace	No namespace
Diagram	<pre> classDiagram class ftp { <<fileFtp>> <<action>> } class action { <<action>> } class fileFtp { <<fileFtp>> <<action>> } fileFtp < -- action fileFtp < -- ftp ftp < -- server ftp < -- port ftp < -- path ftp < -- user ftp < -- pass ftp < -- expiresdatetime server < -- xsd:string port < -- xsd:string path < -- xsd:string user < -- xsd:string pass < -- xsd:string expiresdatetime < -- datetimeGMT </pre> <p>This element contains information about ftp access to file.</p>
Type	fileFtp
Type hierarchy	<ul style="list-style-type: none"> • action • fileFtp
Properties	<p>content: complex</p> <p>minOccurs: 0</p> <p>maxOccurs: 1</p>
Model	ALL(server port path user{0,1} pass{0,1} expiresdatetime)
Children	expiresdatetime, pass, path, port, server, user
Instance	<ftp> <server>{1,1}</server> <port>{1,1}</port> <path>{1,1}</path> <user>{0,1}</user> <pass>{0,1}</pass> <expiresdatetime>{1,1}</expiresdatetime> </ftp>
Source	<xsd:element name="ftp" type="fileFtp" maxOccurs="1" minOccurs="0" />

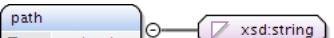
Element fileFtp / server

Namespace	No namespace
Diagram	<p>Built-in primitive type. The string datatype represents character strings in XML.</p>
Type	xsd:string
Properties	content: simple
Source	<xsd:element name="server" type="xsd:string" />

Element fileFtp / port

Namespace	No namespace
Diagram	 <p>Built-in primitive type. The string datatype represents character strings in XML.</p>
Type	xsd:string
Properties	content: simple
Source	<xsd:element name="port" type="xsd:string" />

Element fileFtp / path

Namespace	No namespace
Diagram	 <p>Built-in primitive type. The string datatype represents character strings in XML.</p>
Type	xsd:string
Properties	content: simple
Source	<xsd:element name="path" type="xsd:string" />

Element fileFtp / user

Namespace	No namespace
Diagram	 <p>Built-in primitive type. The string datatype represents character strings in XML.</p>
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="user" type="xsd:string" maxOccurs="1" minOccurs="0" />

Element fileFtp / pass

Namespace	No namespace
Diagram	 <p>Built-in primitive type. The string datatype represents character strings in XML.</p>
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="pass" type="xsd:string" maxOccurs="1" minOccurs="0" />

Element fileFtp / expiresdatetime

Namespace	No namespace
Diagram	
Type	datetimeGMT
Properties	content: simple

Facets	pattern	$\backslash d\{4\}-\backslash d\{2\}-\backslash d\{2\}$ $\backslash d\{2\}:\backslash d\{2\}:\backslash d\{2\}$ GMT\ $+\backslash d\{2\}:\backslash d\{2\}$
Source		<xsd:element name="expiresdatetime" type="datetimeGMT" />

Element file / type

Namespace	No namespace
Diagram	<pre> graph LR type["type
Type fileType"] --> fileType["fileType"] </pre>
Type	fileType
Properties	content: simple minOccurs: 0 maxOccurs: 1
Facets	enumeration full enumeration prelistening enumeration cover enumeration booklet
Source	<xsd:element name="type" type="fileType" maxOccurs="1" minOccurs="0" />

Element file / filetype

Namespace	No namespace
Diagram	<pre> graph LR filetype["filetype
Type xsd:string"] --> xsdString["xsd:string"] </pre> <p>Built-in primitive type. The string datatype represents character strings in XML.</p>
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="filetype" type="xsd:string" maxOccurs="1" minOccurs="0" />

Element file / samplerate

Namespace	No namespace
Diagram	<pre> graph LR samplerate["samplerate
Type xsd:string"] --> xsdString["xsd:string"] </pre> <p>Built-in primitive type. The string datatype represents character strings in XML.</p>
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="samplerate" type="xsd:string" maxOccurs="1" minOccurs="0" />

Element file / samplesize

Namespace	No namespace
Diagram	<pre> graph LR samplesize["samplesize
Type xsd:string"] --> xsdString["xsd:string"] </pre> <p>Built-in primitive type. The string datatype represents character strings in XML.</p>
Type	xsd:string
Properties	content: simple

	minOccurs: 0
	maxOccurs: 1
Source	<xsd:element name="samplesize" type="xsd:string" maxOccurs="1" minOccurs="0" />

Element file / bitrate

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="bitrate" type="xsd:string" maxOccurs="1" minOccurs="0" />

Element file / bitratetype

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="bitratetype" type="xsd:string" maxOccurs="1" minOccurs="0" />

Element file / codec

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="codec" type="xsd:string" maxOccurs="1" minOccurs="0" />

Element file / codecsettings

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="codecsettings" type="xsd:string" maxOccurs="1" minOccurs="0" />

Element file / bytes

Namespace	No namespace
Diagram	
Type	xsd:integer
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<pre><xsd:element name="bytes" type="xsd:integer" maxOccurs="1" minOccurs="0"/></pre>

Element file / checksums

Namespace	No namespace
Diagram	
Type	checksums
Properties	content: complex
Model	ALL(md5{0,1} sha1{0,1} sha256{0,1})
Children	md5, sha1, sha256
Instance	<pre><checksums> <md5>{0,1}</md5> <sha1>{0,1}</sha1> <sha256>{0,1}</sha256> </checksums></pre>
Source	<pre><xsd:element name="checksums" type="checksums"/></pre>

Element checksums / md5

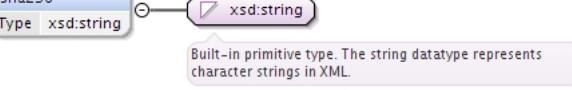
Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<pre><xsd:element name="md5" type="xsd:string" maxOccurs="1" minOccurs="0"/></pre>

Element checksums / sha1

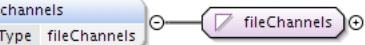
Namespace	No namespace
Diagram	

Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="sha1" type="xsd:string" maxOccurs="1" minOccurs="0" />

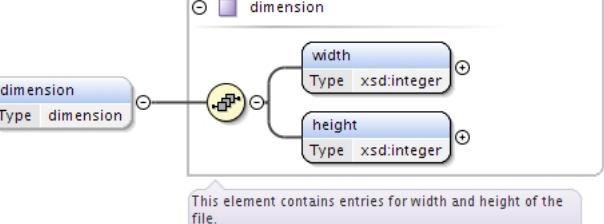
Element checksums / sha256

Namespace	No namespace
Diagram	 <p>The diagram shows a blue rounded rectangle labeled "sha256" with a small "Type" label below it. An arrow points from "sha256" to a purple rounded rectangle labeled "xsd:string". A tooltip below the arrow states: "Built-in primitive type. The string datatype represents character strings in XML."</p>
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="sha256" type="xsd:string" maxOccurs="1" minOccurs="0" />

Element file / channels

Namespace	No namespace
Diagram	 <p>The diagram shows a blue rounded rectangle labeled "channels" with a small "Type" label below it. An arrow points from "channels" to a purple rounded rectangle labeled "fileChannels".</p>
Type	fileChannels
Properties	content: simple minOccurs: 0 maxOccurs: 1
Facets	enumeration mono enumeration stereo enumeration joint-stereo enumeration 5.1
Source	<xsd:element name="channels" type="fileChannels" maxOccurs="1" minOccurs="0" />

Element file / dimension

Namespace	No namespace
Diagram	 <p>The diagram shows a blue rounded rectangle labeled "dimension" with a small "Type" label below it. An arrow points from "dimension" to a purple rounded rectangle labeled "dimension". Inside the "dimension" box, there are two child nodes: "width" (Type: xsd:integer) and "height" (Type: xsd:integer). A tooltip below the "width" node states: "This element contains entries for width and height of the file."</p>
Type	dimension
Properties	content: complex minOccurs: 0 maxOccurs: 1
Model	width , height
Children	height, width
Instance	<dimension> <width>{1,1}</width>

	<pre><height>{1,1}</height> </dimension></pre>
Source	<code><xsd:element name="dimension" type="dimension" maxOccurs="1" minOccurs="0" /></code>

Element dimension / width

Namespace	No namespace
Diagram	
Type	xsd:integer
Properties	content: simple
Source	<code><xsd:element name="width" type="xsd:integer" /></code>

Element dimension / height

Namespace	No namespace
Diagram	
Type	xsd:integer
Properties	content: simple
Source	<code><xsd:element name="height" type="xsd:integer" /></code>

Element file / decryptinfo

Namespace	No namespace
Diagram	
Type	decryptinfo
Properties	content: complex minOccurs: 0 maxOccurs: 1
Model	ALL(cipher{0,1} initvector{0,1} key{0,1} bytes{0,1} checksums{0,1})
Children	bytes, checksums, cipher, initvector, key
Instance	<pre><decryptinfo> <cipher>{0,1}</cipher> <initvector>{0,1}</initvector> <key>{0,1}</key> <bytes>{0,1}</bytes> <checksums>{0,1}</checksums> </decryptinfo></pre>
Source	<code><xsd:element name="decryptinfo" type="decryptinfo" maxOccurs="1" minOccurs="0" /></code>

Element decryptinfo / cipher

Namespace	No namespace
Diagram	<p>The diagram shows the element 'cipher' with its type 'xsd:string'. A callout box indicates that 'xsd:string' is a 'Built-in primitive type. The string datatype represents character strings in XML.'</p>
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="cipher" type="xsd:string" minOccurs="0" maxOccurs="1"/>

Element decryptinfo / initvector

Namespace	No namespace
Diagram	<p>The diagram shows the element 'initvector' with its type 'xsd:string'. A callout box indicates that 'xsd:string' is a 'Built-in primitive type. The string datatype represents character strings in XML.'</p>
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="initvector" type="xsd:string" minOccurs="0" maxOccurs="1"/>

Element decryptinfo / key

Namespace	No namespace
Diagram	<p>The diagram shows the element 'key' with its type 'xsd:string'. A callout box indicates that 'xsd:string' is a 'Built-in primitive type. The string datatype represents character strings in XML.'</p>
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="key" type="xsd:string" minOccurs="0" maxOccurs="1"/>

Element decryptinfo / bytes

Namespace	No namespace
Diagram	<p>The diagram shows the element 'bytes' with its type 'xsd:string'. A callout box indicates that 'xsd:string' is a 'Built-in primitive type. The string datatype represents character strings in XML.'</p>
Type	xsd:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<xsd:element name="bytes" type="xsd:string" minOccurs="0" maxOccurs="1"/>

Element decryptinfo / checksums

Namespace	No namespace
-----------	--------------

Diagram	
Type	checksums
Properties	<p>content: complex</p> <p>minOccurs: 0</p> <p>maxOccurs: 1</p>
Model	ALL(md5{0,1} sha1{0,1} sha256{0,1})
Children	md5, sha1, sha256
Instance	<pre><checksums> <md5>{0,1}</md5> <sha1>{0,1}</sha1> <sha256>{0,1}</sha256> </checksums></pre>
Source	<code><xsd:element name="checksums" type="checksums" minOccurs="0" maxOccurs="1"/></code>

Element bundle / purchase

Namespace	No namespace
Diagram	
Type	purchase
Properties	<p>content: complex</p> <p>minOccurs: 0</p> <p>maxOccurs: 1</p>
Model	ALL(pos url)
Children	pos, url
Instance	<pre><purchase> <pos>{1,1}</pos> <url>{1,1}</url> </purchase></pre>
Source	<code><xsd:element name="purchase" type="purchase" maxOccurs="1" minOccurs="0"/></code>

Element purchase / pos

Namespace	No namespace
Diagram	
Type	xsd:string
Properties	content: simple
Source	<code><xsd:element name="pos" type="xsd:string"/></code>

Element purchase / url

Namespace	No namespace
Diagram	<p>The diagram shows the 'url' element with a type of 'xsd:string'. A callout bubble indicates it is a built-in primitive type representing character strings in XML.</p>
Type	xsd:string
Properties	content: simple
Source	<xsd:element name="url" type="xsd:string"/>

Element item / license_basis

Namespace	No namespace
Diagram	<p>The diagram shows the 'license_basis' element with several child elements: territorial, timeframe, pricing, streaming_allowed, channels, and as_on_bundle. A callout bubble indicates this element includes basic rules and information under which the bundle is provided.</p>
Type	license_basis
Properties	content: complex
Model	ALL(territorial{0,1} timeframe{0,1} pricing{0,1} streaming_allowed{0,1} channels{0,1} as_on_bundle{0,1})
Children	as_on_bundle, channels, pricing, streaming_allowed, territorial, timeframe
Instance	<pre><license_basis> <territorial>{0,1}</territorial> <timeframe>{0,1}</timeframe> <pricing>{0,1}</pricing> <streaming_allowed>{0,1}</streaming_allowed> <channels>{0,1}</channels> <as_on_bundle>{0,1}</as_on_bundle> </license_basis></pre>
Source	<xsd:element name="license_basis" type="license_basis"/>

Element item / license_specifics

Namespace	No namespace
Diagram	<p>The diagram shows the 'license_specifics' element with a child element 'rules'. A callout bubble indicates this element includes specific rules which should be applied.</p>
Type	license_specifics
Properties	content: complex
Model	rules
Children	rules

Instance	<pre><license_specifics> <rules>{1,1}</rules> </license_specifics></pre>
Source	<pre><xsd:element name="license_specifics" type="license_specifics"/></pre>

Element item / tags

Namespace	No namespace						
Diagram	<pre> classDiagram class tags { genres bundle_only explicit_lyrics live acoustic instrumental } tags < --> "2..5" tags </pre> <p>This element contains information about genres and more.</p>						
Type	tags						
Properties	<table border="1"> <tr> <td>content:</td> <td>complex</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	complex	minOccurs:	0	maxOccurs:	1
content:	complex						
minOccurs:	0						
maxOccurs:	1						
Model	ALL(genres{0,1} bundle_only{0,1} explicit_lyrics{0,1} live{0,1} acoustic{0,1} instrumental{0,1})						
Children	acoustic, bundle_only, explicit_lyrics, genres, instrumental, live						
Instance	<pre> <tags> <genres>{0,1}</genres> <bundle_only>{0,1}</bundle_only> <explicit_lyrics>{0,1}</explicit_lyrics> <live>{0,1}</live> <acoustic>{0,1}</acoustic> <instrumental>{0,1}</instrumental> </tags> </pre>						
Source	<pre><xsd:element name="tags" type="tags" maxOccurs="1" minOccurs="0" /></pre>						

Element item / fingerprint

Namespace	No namespace						
Diagram	<pre> classDiagram class fingerprint { echoprint } fingerprint < --> "2..5" fingerprint </pre> <p>This element includes an element "echoprint" (http://echoprint.me https://github.com/echonest/echoprint-codegen).</p>						
Type	fingerprint						
Properties	<table border="1"> <tr> <td>content:</td> <td>complex</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	complex	minOccurs:	0	maxOccurs:	1
content:	complex						
minOccurs:	0						
maxOccurs:	1						
Model	echoprint{0,1}						
Children	echoprint						
Instance	<pre> <fingerprint> <echoprint>{0,1}</echoprint> </fingerprint> </pre>						

Source	<code><xsd:element name="fingerprint" type="fingerprint" maxOccurs="1" minOccurs="0" /></code>
--------	--

Element **fingerprint** / **echoprint**

Namespace	No namespace						
Diagram	<p>The diagram shows a box labeled "echoprint" with a "Type" label "xsd:string". An arrow points from "echoprint" to "xsd:string". A callout bubble states: "Built-in primitive type. The string datatype represents character strings in XML."</p>						
Type	xsd:string						
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	simple	minOccurs:	0	maxOccurs:	1
content:	simple						
minOccurs:	0						
maxOccurs:	1						
Source	<code><xsd:element name="echoprint" type="xsd:string" maxOccurs="1" minOccurs="0" /></code>						

Element **item** / **reporting**

Namespace	No namespace						
Diagram	<p>The diagram shows a box labeled "reporting" with a "Type" label "reporting". Inside "reporting", there are two boxes: "realtime" and "postponed", each with a "Type" label "realtime" and "postponed" respectively. Arrows point from "reporting" to both "realtime" and "postponed". A callout bubble states: "This element contains information about reporting."</p>						
Type	reporting						
Properties	<table border="1"> <tr> <td>content:</td> <td>complex</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	complex	minOccurs:	0	maxOccurs:	1
content:	complex						
minOccurs:	0						
maxOccurs:	1						
Model	ALL(realtime postponed)						
Children	postponed, realtime						
Instance	<code><reporting> <realtime>{1,1}</realtime> <postponed>{1,1}</postponed> </reporting></code>						
Source	<code><xsd:element name="reporting" type="reporting" maxOccurs="1" minOccurs="0" /></code>						

Element **item** / **files**

Namespace	No namespace						
Diagram	<p>The diagram shows a box labeled "files" with a "Type" label "files". Inside "files", there is a box labeled "file" with a "Type" label "file". An arrow points from "files" to "file". A callout bubble states: "This element contains a list of files."</p>						
Type	files						
Properties	<table border="1"> <tr> <td>content:</td> <td>complex</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	complex	minOccurs:	0	maxOccurs:	1
content:	complex						
minOccurs:	0						
maxOccurs:	1						
Model	file*						
Children	file						
Instance	<code><files> <file>{0,unbounded}</file> </files></code>						
Source	<code><xsd:element name="files" type="files" maxOccurs="1" minOccurs="0" /></code>						

Element feed / item

Namespace	No namespace						
Diagram	<p>This element contains information about a item just like a track. The type describes what the item is e.g. audio,...</p>						
Type	item						
Properties	<table border="1"> <tr> <td>content:</td><td>complex</td></tr> <tr> <td>minOccurs:</td><td>0</td></tr> <tr> <td>maxOccurs:</td><td>unbounded</td></tr> </table>	content:	complex	minOccurs:	0	maxOccurs:	unbounded
content:	complex						
minOccurs:	0						
maxOccurs:	unbounded						
Model	ALL(displayname name version type display_artistname{0,1} ids{0,1} contributors information license_basis license_specifics tags{0,1} fingerprint{0,1} reporting{0,1} files{0,1})						
Children	contributors, display_artistname, displayname, files, fingerprint, ids, information, license_basis, license_specifics, name, reporting, tags, type, version						
Instance	<pre> <item> <displayname>{1,1}</displayname> <name>{1,1}</name> <version>{1,1}</version> <type>{1,1}</type> <display_artistname>{0,1}</display_artistname> <ids>{0,1}</ids> <contributors>{1,1}</contributors> <information>{1,1}</information> <license_basis>{1,1}</license_basis> <license_specifics>{1,1}</license_specifics> <tags>{0,1}</tags> <fingerprint>{0,1}</fingerprint> <reporting>{0,1}</reporting> <files>{0,1}</files> </item> </pre>						

Source

```
<xsd:element name="item" type="item" maxOccurs="unbounded" minOccurs="0"/>
```

Complex Type(s)

Complex Type feedinfo

Namespace	No namespace
Annotations	<p>On feedinfo-level there are the global information needed or at least valuable for ingesting / identifying the content sent. It is defined, when the feed was created, when it shall be come effective, who created the feed and who is the receiver of the feed. Also the sender (which can diverge from the creator) is to be stated. The licensor is also to be stated (which in turn can also diverge from the creator and/or the sender). There can be "actions" defined on the receiving-party's side which should be "done" when initially receiving this feed, or starting to process the feed for ingestion or finishing the feeds processing. Additionally when everything could be interpreted correctly (in the sense of the receiving party), a "full-success-action" could be issued; likewise if "some error" occurred while processing the feed, an "onerror-action" could be issued. Those actions are initially defined to be email-notifications or http-calls; we also included some action to have a "registered letter" and/or "fax" to be sent; wether this is accepted/handled by the receiving party is to be dealt with contractually (we included a field for stating how much the sending party will cover the fee max.).</p>
Diagram	<pre> classDiagram class feedinfo { onlytest :xsd:boolean feedid :xsd:string creationdatetime :datetimeGMT effectivedatetime :datetimeGMT creator :creator receiver :receiver sender :sender licensor :licensor licensee :licensee actions :actions } </pre>
Used by	Element feed/feedinfo
Model	ALL(onlytest feedid creationdatetime effectivedatetime creator receiver sender licensor licensee actions{0,1})
Children	actions, creationdatetime, creator, effectivedatetime, feedid, licensee, licensor, onlytest, receiver, sender
Source	<pre> <xsd:complexType name="feedinfo"> <xsd:annotation> <xsd:documentation xml:lang="en">On feedinfo-level there are the global information needed or at least valuable for ingesting / identifying the content sent. It is defined, when the feed was created, when it shall be come effective, who created the feed and who is the receiver of the feed. Also the sender (which can diverge from the creator) is to be stated. The licensor is also to be stated (which in turn can also diverge from the creator and/or the sender). There can be "actions" defined on the receiving-party's side which should be "done" when initially receiving this feed, or starting to process the feed for ingestion or finishing the feeds processing. Additionally when everything could be interpreted correctly (in the sense of the receiving party), a "full-success-action" could be issued; likewise if "some error" occurred while processing the feed,</pre>

an "onerror-action" could be issued. Those actions are initially defined to be email-notifications or http-calls; we also included some action to have a "registered letter" and/or "fax" to be sent; whether this is accepted/handled by the receiving party is to be dealt with contractually (we included a field for stating how much the sending party will cover the fee max.).</xsd:documentation>

```
</xsd:annotation>
<xsd:all>
  <xsd:element name="onlytest" type="xsd:boolean"/>
  <xsd:element name="feedid" type="xsd:string"/>
  <xsd:element name="creationdatetime" type="datetimeGMT"/>
  <xsd:element name="effectivedatetime" type="datetimeGMT"/>
  <xsd:element name="creator" type="creator"/>
  <xsd:element name="receiver" type="receiver"/>
  <xsd:element name="sender" type="sender"/>
  <xsd:element name="licensor" type="licensor"/>
  <xsd:element name="licensee" type="licensee"/>
  <xsd:element name="actions" type="actions" maxOccurs="1" minOccurs="0"/>
</xsd:all>
</xsd:complexType>
```

Complex Type creator

Namespace	No namespace
Annotations	This element contains information about the creator of that feed.
Diagram	<pre> classDiagram class creator { email : Type email userid : Type userid keyid : Type xsd:string } </pre> <p>This element contains information about the creator of that feed.</p>
Used by	Element feedinfo/creator
Model	ALL(email userid keyid)
Children	email, keyid, userid
Source	<pre> <xsd:complexType name="creator"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information about the creator of that feed.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="email" type="email"/> <xsd:element name="userid" type="userid"/> <xsd:element name="keyid" type="xsd:string"/> </xsd:all> </xsd:complexType></pre>

Complex Type receiver

Namespace	No namespace
Annotations	This element contains information about the receiver of that feed.
Diagram	<pre> classDiagram class receiver { type : Type receivertypes servername : Type iporhostname serveripv4 : Type ipv4 serveripv6 : Type ipv6 authtype : Type authtype username : Type xsd:string crypto : Type crypto } </pre> <p>This element contains information about the receiver of that feed.</p>

Used by	Element feedinfo/receiver
Model	ALL(type servername serveripv4 serveripv6 authtype username crypto)
Children	authtype, crypto, serveripv4, serveripv6, servername, type, username
Source	<pre><xsd:complexType name="receiver"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information about the receiver of that feed.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="type" type="receivertypes"/> <xsd:element name="servername" type="iporhostname"/> <xsd:element name="serveripv4" type="ipv4"/> <xsd:element name="serveripv6" type="ipv6"/> <xsd:element name="authtype" type="authtype"/> <xsd:element name="username" type="xsd:string"/> <xsd:element name="crypto" type="crypto"/> </xsd:all> </xsd:complexType></pre>

Complex Type crypto

Namespace	No namespace
Annotations	This element contains crypto information for secure and authenticated transfer.
Diagram	<p>This element contains crypto information for secure and authenticated transfer.</p>
Used by	Element receiver/crypto
Model	ALL(relatedemail usedkeyid usedpubkey)
Children	relatedemail, usedkeyid, usedpubkey
Source	<pre><xsd:complexType name="crypto"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains crypto information for secure and authenticated transfer.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="relatedemail" type="email"/> <xsd:element name="usedkeyid" type="keyid"/> <xsd:element name="usedpubkey" type="xsd:base64Binary"/> </xsd:all> </xsd:complexType></pre>

Complex Type sender

Namespace	No namespace
Annotations	This element contains information about the sender of that feed.
Diagram	<p>This element contains information about the sender of that feed.</p>
Used by	Element feedinfo/sender
Model	ALL(contractpartnerid ourcontractpartnerid email keyid)
Children	contractpartnerid, email, keyid, ourcontractpartnerid
Source	<pre><xsd:complexType name="sender"> <xsd:annotation></pre>

```

<xsd:documentation xml:lang="en">This element contains information about the sender of
that feed.</xsd:documentation>
</xsd:annotation>
<xsd:all>
  <xsd:element name="contractpartnerid" type="xsd:string"/>
  <xsd:element name="ourcontractpartnerid" type="xsd:string"/>
  <xsd:element name="email" type="email"/>
  <xsd:element name="keyid" type="xsd:string"/>
</xsd:all>
</xsd:complexType>
```

Complex Type licensor

Namespace	No namespace
Annotations	This element contains information about the licensor of that feed.
Diagram	<pre> classDiagram class licensor { contractpartnerid : xsd:string ourcontractpartnerid : xsd:string email : email keyid : xsd:string } </pre> <p>This element contains information about the licensor of that feed.</p>
Used by	Element feedinfo/licensor
Model	ALL(contractpartnerid ourcontractpartnerid email keyid)
Children	contractpartnerid, email, keyid, ourcontractpartnerid
Source	<pre> <xsd:complexType name="licensor"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information about the licensor of that feed.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="contractpartnerid" type="xsd:string"/> <xsd:element name="ourcontractpartnerid" type="xsd:string"/> <xsd:element name="email" type="email"/> <xsd:element name="keyid" type="xsd:string"/> </xsd:all> </xsd:complexType></pre>

Complex Type licensee

Namespace	No namespace
Annotations	This element contains information about the licensee of that feed.
Diagram	<pre> classDiagram class licensee { contractpartnerid : xsd:string ourcontractpartnerid : xsd:string email : email keyid : xsd:string } </pre> <p>This element contains information about the licensee of that feed.</p>
Used by	Element feedinfo/licensee
Model	ALL(contractpartnerid ourcontractpartnerid email keyid)
Children	contractpartnerid, email, keyid, ourcontractpartnerid
Source	<pre> <xsd:complexType name="licensee"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information about the licensee of that feed.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="contractpartnerid" type="xsd:string"/> <xsd:element name="ourcontractpartnerid" type="xsd:string"/> <xsd:element name="email" type="email"/> </xsd:all> </xsd:complexType></pre>

```
<xsd:element name="keyid" type="xsd:string"/>
</xsd:all>
</xsd:complexType>
```

Complex Type actions

Namespace	No namespace
Annotations	This element contains information about possible actions with the feed.
Diagram	<p>This element contains information about possible actions with the feed.</p>
Used by	Element feedinfo/actions
Model	ALL(oninitialreceive onprocessstart onprocessend onfullsuccess onerror)
Children	onerror, onfullsuccess, oninitialreceive, onprocessend, onprocessstart
Source	<pre><xsd:complexType name="actions"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information about possible actions with the feed.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="oninitialreceive" type="event" /> <xsd:element name="onprocessstart" type="event" /> <xsd:element name="onprocessend" type="event" /> <xsd:element name="onfullsuccess" type="event" /> <xsd:element name="onerror" type="event" /> </xsd:all> </xsd:complexType></pre>

Complex Type event

Namespace	No namespace
Annotations	This element contains information about possible events and actions.
Diagram	<p>This element contains information about possible events and actions.</p>
Used by	Elements actions/onerror, actions/onfullsuccess, actions/oninitialreceive, actions/onprocessend, actions/onprocessstart Complex Types onerror, onfullsuccess, oninitialreceive, onprocessend, onprocessstart
Model	mailto*, http*, fax*, letter*
Children	fax, http, letter, mailto
Source	<pre><xsd:complexType name="event"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information about possible events and actions.</xsd:documentation> </xsd:annotation> <xsd:sequence> <xsd:element name="mailto" type="mailto" minOccurs="0" maxOccurs="unbounded" /> <xsd:element name="http" type="http" minOccurs="0" maxOccurs="unbounded" /> <xsd:element name="fax" type="fax" minOccurs="0" maxOccurs="unbounded" /> <xsd:element name="letter" type="letter" minOccurs="0" maxOccurs="unbounded" /> </xsd:sequence> </xsd:complexType></pre>

```
</xsd:sequence>
</xsd:complexType>
```

Complex Type mailto

Namespace	No namespace
Annotations	This element contains information about mailto-event.
Diagram	<pre> classDiagram class mailto { <<Base Type action>> <<This element contains information about mailto-event.>> receiver subject text } class action { <<action (extension base)>> } mailto < -- action mailto "1..oo" --> receiver mailto --> subject mailto --> text </pre>
Type	extension of action
Type hierarchy	<ul style="list-style-type: none"> • action • mailto
Used by	Element event/mailto
Model	receiver+, subject, text
Children	receiver, subject, text
Source	<pre> <xsd:complexType name="mailto"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information about mailto- event.</xsd:documentation> </xsd:annotation> <xsd:complexContent> <xsd:extension base="action"> <xsd:sequence> <xsd:element name="receiver" type="emaillist" minOccurs="1" maxOccurs="unbounded" /> <xsd:element name="subject" type="xsd:string" /> <xsd:element name="text" type="xsd:string" /> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType> </pre>

Complex Type action

Namespace	No namespace
Diagram	<pre> classDiagram class http { <<Base Type action>> <<This element contains information about http-event.>> url type addheader addparams } class action { <<action (extension base)>> } http < -- action http --> url http --> type http --> addheader http --> addparams </pre>
Used by	Complex Types fax, fileFtp, fileHttp, http, mailto
Source	<pre> <xsd:complexType name="action"> ... </xsd:complexType> </pre>

Complex Type http

Namespace	No namespace
Annotations	This element contains information about http-event.
Diagram	<pre> classDiagram class http { <<Base Type action>> <<This element contains information about http-event.>> url type addheader addparams } class action { <<action (extension base)>> } http < -- action http --> url http --> type http --> addheader http --> addparams </pre>

Type	extension of action
Type hierarchy	<ul style="list-style-type: none"> • action <ul style="list-style-type: none"> • http
Used by	Elements event/http, realtime/http
Model	ALL(url type addheader addparams)
Children	addheader, addparams, type, url
Source	<pre><xsd:complexType name="http"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information about http-event.</ xsd:documentation> </xsd:annotation> <xsd:complexContent> <xsd:extension base="action"> <xsd:all> <xsd:element name="url" type="url"/> <xsd:element name="type" type="httpmethods"/> <xsd:element name="addheader" type="http_addheader"/> <xsd:element name="addparams" type="http_addparams"/> </xsd:all> </xsd:extension> </xsd:complexContent> </xsd:complexType></pre>

Complex Type http_addheader

Namespace	No namespace
Diagram	<pre> classDiagram class http_addheader { <<Base Type>> } class action_instruction { <<extension base>> } http_addheader < -- action_instruction <#any> *-- 1..∞ : http_addheader </pre>
Type	extension of action_instruction
Type hierarchy	<ul style="list-style-type: none"> • action_instruction <ul style="list-style-type: none"> • http_addheader
Used by	Element http/addheader
Model	ANY element from ANY namespace
Source	<pre><xsd:complexType name="http_addheader"> <xsd:complexContent mixed="false"> <xsd:extension base="action_instruction"> <xsd:sequence> <xsd:any processContents="lax" maxOccurs="unbounded"/> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType></pre>

Complex Type action_instruction

Namespace	No namespace
Diagram	<pre> classDiagram class action_instruction </pre>
Used by	Complex Types http_addheader, http_addparams
Source	<pre><xsd:complexType name="action_instruction">/></pre>

Complex Type http_addparams

Namespace	No namespace
Diagram	<pre> classDiagram class http_addparams { <<Base Type>> } class action_instruction { <<extension base>> } http_addparams < -- action_instruction <#any> *-- 1..∞ : http_addparams </pre>
Type	extension of action_instruction
Type hierarchy	<ul style="list-style-type: none"> • action_instruction

	<ul style="list-style-type: none"> http_addparams
Used by	Element http/addparams
Model	ANY element from ANY namespace
Source	<pre><xsd:complexType name="http_addparams"> <xsd:complexContent> <xsd:extension base="action_instruction"> <xsd:sequence> <xsd:any processContents="lax" maxOccurs="unbounded" /> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType></pre>

Complex Type fax

Namespace	No namespace
Diagram	<pre> classDiagram fax "Base Type" --o action fax "1..*" --o "#any" action "*" --o "#any" </pre>
Type	extension of action
Type hierarchy	<ul style="list-style-type: none"> action fax
Used by	Element event/fax
Model	ANY element from ANY namespace
Source	<pre><xsd:complexType name="fax"> <xsd:complexContent> <xsd:extension base="action"> <xsd:sequence> <xsd:any processContents="lax" maxOccurs="unbounded" /> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType></pre>

Complex Type letter

Namespace	No namespace
Annotations	This element contains information about the letter event.
Diagram	<pre> classDiagram letter "*" --o registered "Type xsd:boolean" letter "*" --o to "Type xsd:string" letter "*" --o text "Type xsd:string" letter "*" --o costscoveredby "Type xsd:string" note over letter: This element contains information about the letter event. </pre>
Used by	Element event/letter
Model	ALL(registered to text costscoveredby)
Children	costscoveredby, registered, text, to
Source	<pre><xsd:complexType name="letter"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information about the letter event.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="registered" type="xsd:boolean" /> <xsd:element name="to" type="xsd:string" /> <xsd:element name="text" type="xsd:string" /> <xsd:element name="costscoveredby" type="xsd:string" /> </xsd:all> </xsd:complexType></pre>

Complex Type to

Namespace	No namespace
Annotations	This element contains information about recipient.
Diagram	<pre> classDiagram class to { name : xsd:string department : xsd:string nameperson : xsd:string street : xsd:string postcode : xsd:string country : xsd:string additionaladdressinfo : xsd:string } note over to: This element contains information about recipient. </pre>
Used by	Element letter/to
Model	ALL(name{0,1} department{0,1} nameperson{0,1} street postcode country additionaladdressinfo{0,1})
Children	additionaladdressinfo, country, department, name, nameperson, postcode, street
Source	<pre> <xsd:complexType name="to"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information about recipient.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="name" type="xsd:string" minOccurs="0" maxOccurs="1"/> <xsd:element name="department" type="xsd:string" minOccurs="0" maxOccurs="1"/> <xsd:element name="nameperson" type="xsd:string" minOccurs="0" maxOccurs="1"/> <xsd:element name="street" type="xsd:string"/> <xsd:element name="postcode" type="xsd:string"/> <xsd:element name="country" type="xsd:string"/> <xsd:element name="additionaladdressinfo" type="xsd:string" minOccurs="0" maxOccurs="1"/> </xsd:all> </xsd:complexType> </pre>

Complex Type costscoveredby

Namespace	No namespace
Annotations	This element contains information about who covered the costs of event.
Diagram	<pre> classDiagram class costscoveredby { contractpartnerid : xsd:string ourcontractpartnerid : xsd:string maxcostscovered : xsd:string } note over costscoveredby: This element contains information about who covered the costs of event. </pre>
Used by	Element letter/costscoveredby
Model	ALL(contractpartnerid ourcontractpartnerid maxcostscovered{0,1})
Children	contractpartnerid, maxcostscovered, ourcontractpartnerid
Source	<pre> <xsd:complexType name="costscoveredby"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information about who covered the costs of event.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="contractpartnerid" type="xsd:string"/> <xsd:element name="ourcontractpartnerid" type="xsd:string"/> <xsd:element name="maxcostscovered" type="xsd:string" minOccurs="0" maxOccurs="1"/> </xsd:all> </xsd:complexType> </pre>

Complex Type bundle

Namespace	No namespace
Annotations	<p>On bundle level, there are information on how to handle a collection of "items". This is mainly an album/ep/single. A bundle is identified by one unique identifier, but more unique identifiers could and should be transmitted as well (see below "ids"). Most notably on the bundle-level is the "bundle name" which is basically the conjunction of the "name"- and the "version"-field. Also to have this easy at hand, there should be the desired "display_artistname"-string be present on this level. Of course, the receiver of the feed can still calculate the "correct" display_artistname by evaluating the contributors (see below) for this.</p>
Diagram	<pre> classDiagram class bundle { +displayname +name +version +display_artistname +ids +items +contributors +information +license_basis +license_specifics +reporting +tags +files +purchase } bundle --> bundle : + </pre>
Used by	Elements feed/bundle, related/bundle
Model	ALL(displayname{0,1} name{0,1} version{0,1} display_artistname{0,1} ids items{0,1} contributors{0,1} information{0,1} license_basis{0,1} license_specifics{0,1} reporting{0,1} tags{0,1} files{0,1} purchase{0,1})
Children	contributors, display_artistname, displayname, files, ids, information, items, license_basis, license_specifics, name, purchase, reporting, tags, version
Source	<pre> <xsd:complexType name="bundle"> <xsd:annotation> <xsd:documentation xml:lang="en">On bundle level, there are information on how to handle a collection of "items". This is mainly an album/ep/single. A bundle is identified by one unique identifier, but more unique identifiers could and should be transmitted as well (see below "ids"). Most notably on the bundle-level is the "bundle name" which is basically the conjunction of the "name"- and the "version"-field. Also to have this easy at hand, there should be the desired "display_artistname"-string be present on this level. Of course, the receiver of the feed can still calculate the "correct" display_artistname by evaluating the contributors (see below) for this.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="displayname" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="name" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="version" type="xsd:string" maxOccurs="1" minOccurs="0"/> </pre>

```

<xsd:element name="display_artistname" type="xsd:string" maxOccurs="1" minOccurs="0"/>
<xsd:element name="ids" type="ids"/>
<xsd:element name="items" type="items" maxOccurs="1" minOccurs="0"/>
<xsd:element name="contributors" type="contributors" maxOccurs="1" minOccurs="0"/>
<xsd:element name="information" type="information" maxOccurs="1" minOccurs="0"/>
<xsd:element name="license_basis" type="license_basis" maxOccurs="1" minOccurs="0"/>
<xsd:element name="license_specifics" type="license_specifics" maxOccurs="1"
minOccurs="0"/>
<xsd:element name="reporting" type="reporting" maxOccurs="1" minOccurs="0"/>
<xsd:element name="tags" type="tags" maxOccurs="1" minOccurs="0"/>
<xsd:element name="files" type="files" maxOccurs="1" minOccurs="0"/>
<xsd:element name="purchase" type="purchase" maxOccurs="1" minOccurs="0"/>
</xsd:all>
</xsd:complexType>

```

Complex Type ids

Namespace	No namespace
Annotations	This Element is a container for all IDs which are available for the associated element.
Diagram	<pre> graph TD ids[ids] -- "0..1" --> grid ids -- "0..1" --> upc ids -- "0..1" --> isrc ids -- "0..1" --> contentauth ids -- "0..1" --> labelordernum ids -- "0..1" --> amzn ids -- "0..1" --> isbn ids -- "0..1" --> finetunes ids -- "0..1" --> licensor ids -- "0..1" --> licensee ids -- "0..1" --> gvl </pre>
Used by	Elements bundle/ids, contributor/ids, item/ids
Model	ALL(grid{0,1} upc{0,1} isrc{0,1} contentauth{0,1} labelordernum{0,1} amzn{0,1} isbn{0,1} finetunes{0,1} licensor{0,1} licensee{0,1} gvl{0,1})
Children	amzn, contentauth, finetunes, grid, gvl, isbn, isrc, labelordernum, licensee, licensor, upc
Source	<pre> <xsd:complexType name="ids"> <xsd:annotation> <xsd:documentation xml:lang="en">This Element is a container for all IDs which are available for the associated element.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="grid" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="upc" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="isrc" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="contentauth" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="labelordernum" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="amzn" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="isbn" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="finetunes" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="licensor" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="licensee" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="gvl" type="xsd:string" maxOccurs="1" minOccurs="0"/> </xsd:all> </xsd:complexType> </pre>

Complex Type items

Namespace	No namespace
Annotations	This element is a container for item-elements.
Diagram	<pre> classDiagram class items { <<This element is a container for item-elements.>> } class item { <<Type item>> } items "1..*" --> "1..1" item </pre>
Used by	Element bundle/items
Model	item+
Children	item
Source	<pre> <xsd:complexType name="items"> <xsd:annotation> <xsd:documentation xml:lang="en">This element is a container for item-elements.</xsd:documentation> </xsd:annotation> <xsd:sequence> <xsd:element name="item" type="item" maxOccurs="unbounded" minOccurs="1"/> </xsd:sequence> </xsd:complexType> </pre>

Complex Type item

Namespace	No namespace
Annotations	<p>This element contains information about a item just like a track. The type describes what the item is e.g. audio, video, android-app et cetera. The entry "version" is important if different versions of the bundle exist. The licens_basic and license_specifics contains information and rules about pricing, allowed and disallowed territories, channels an so on. The child "files" hold information for the associated files for this item.</p>

Diagram	<pre> classDiagram class item { displayname name version type display_artistname ids contributors information license_basis license_specifics tags fingerprint reporting files } item "1..*" --> "1..*" item </pre>
Used by	Elements feed/item, items/item
Model	ALL(displayname name version type display_artistname{0,1} ids{0,1} contributors information license_basis license_specifics tags{0,1} fingerprint{0,1} reporting{0,1} files{0,1})
Children	contributors, display_artistname, displayname, files, fingerprint, ids, information, license_basis, license_specifics, name, reporting, tags, type, version
Source	<pre> <xsd:complexType name="item"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information about a item just like a track. The type describes what the item is e.g. audio, video, android-app et cetera. The entry "version" is important if different versions of the bundle exist. The licens_basic and license_specifics contains information and rules about pricing, allowed and disallowed territories, channels an so on. The child "files" hold information for the associated files for this item.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="displayname" type="xsd:string"/> <xsd:element name="name" type="xsd:string"/> <xsd:element name="version" type="xsd:string"/> <xsd:element name="type" type="xsd:string"/> <xsd:element name="display_artistname" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="ids" type="ids" maxOccurs="1" minOccurs="0"/> <xsd:element name="contributors" type="contributors"/> <xsd:element name="information" type="information"/> <xsd:element name="license_basis" type="license_basis"/> <xsd:element name="license_specifics" type="license_specifics"/> <xsd:element name="tags" type="tags" maxOccurs="1" minOccurs="0"/> <xsd:element name="fingerprint" type="fingerprint" maxOccurs="1" minOccurs="0"/> <xsd:element name="reporting" type="reporting" maxOccurs="1" minOccurs="0"/> <xsd:element name="files" type="files" maxOccurs="1" minOccurs="0"/> </xsd:all> </xsd:complexType> </pre>

Complex Type contributors

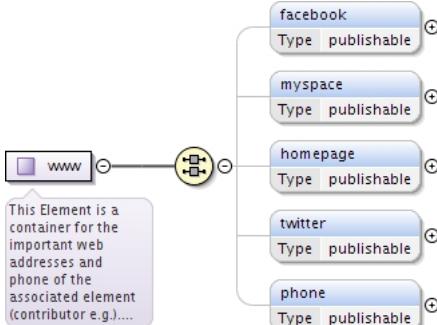
Namespace	No namespace
-----------	--------------

Annotations	This element contains a list of contributor.
Diagram	<pre> classDiagram class contributors class contributor contributors "1..>" contributor </pre>
Used by	Elements bundle/contributors, item/contributors
Model	contributor+
Children	contributor
Source	<pre> <xsd:complexType name="contributors"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains a list of contributor.</xsd:documentation> </xsd:annotation> <xsd:sequence> <xsd:element name="contributor" type="contributor" maxOccurs="unbounded" minOccurs="1"/> </xsd:sequence> </xsd:complexType> </pre>

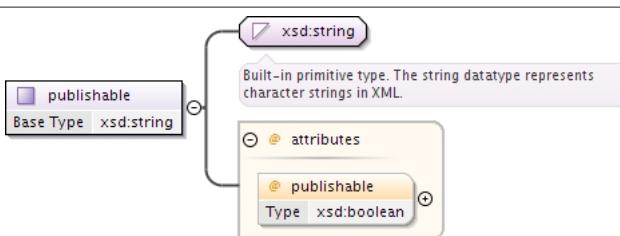
Complex Type contributor

Namespace	No namespace										
Annotations	<p>This element contains information of one contributor. A contributor can be a label, performer, texter, writer, vocals, conductor, display_artist, compiler, copyright, production or clearinghouse. A Year should be provided in case the type equals copyright or production.</p>										
Diagram	<pre> classDiagram class contributor { @ num :xsd:integer name :xsd:string type :contributorType year :xsd:string ids :ids www :www } </pre>										
Used by	Element contributors/contributor										
Model	ALL(name type year{0,1} ids www{0,1})										
Children	ids, name, type, www, year										
Attributes	<table border="1"> <thead> <tr> <th>QName</th> <th>Type</th> <th>Fixed</th> <th>Default</th> <th>Use</th> </tr> </thead> <tbody> <tr> <td>num</td> <td>xsd:integer</td> <td></td> <td></td> <td>optional</td> </tr> </tbody> </table>	QName	Type	Fixed	Default	Use	num	xsd:integer			optional
QName	Type	Fixed	Default	Use							
num	xsd:integer			optional							
Source	<pre> <xsd:complexType name="contributor"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information of one contributor. A contributor can be a label, performer, texter, writer, vocals, conductor, display_artist, compiler, copyright, production or clearinghouse. A Year should be provided in case the type equals copyright or production.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="name" type="xsd:string"/> <xsd:element name="type" type="contributorType"/> <xsd:element name="year" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="ids" type="ids"/> <xsd:element name="www" type="www" maxOccurs="1" minOccurs="0"/> </xsd:all> <xsd:attribute name="num" type="xsd:integer"/> </xsd:complexType> </pre>										

Complex Type www

Namespace	No namespace
Annotations	<p>This Element is a container for the important web addresses and phone of the associated element (contributor e.g.). Phone should be in international format.</p> <p>Every single information-entry cold be tagged "publishable" which would then mean wether customers of receiver are also allowed to be given this information. If publishable is not given, then this is granted.</p>
Diagram	 <pre> classDiagram www < -- facebook www < -- myspace www < -- homepage www < -- twitter www < -- phone www --> Note: This Element is a container for the important web addresses and phone of the associated element (contributor e.g.) facebook --> Note: Type publishable myspace --> Note: Type publishable homepage --> Note: Type publishable twitter --> Note: Type publishable phone --> Note: Type publishable </pre>
Used by	Element contributor/www
Model	ALL/facebook{0,1} myspace{0,1} homepage{0,1} twitter{0,1} phone{0,1})
Children	facebook, homepage, myspace, phone, twitter
Source	<pre> <xsd:complexType name="www"> <xsd:annotation> <xsd:documentation xml:lang="en">This Element is a container for the important web addresses and phone of the associated element (contributor e.g.). Phone should be in international format. Every single information-entry cold be tagged "publishable" which would then mean wether customers of receiver are also allowed to be given this information. If publishable is not given, then this is granted.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="facebook" type="publishable" maxOccurs="1" minOccurs="0"/> <xsd:element name="myspace" type="publishable" maxOccurs="1" minOccurs="0"/> <xsd:element name="homepage" type="publishable" maxOccurs="1" minOccurs="0"/> <xsd:element name="twitter" type="publishable" maxOccurs="1" minOccurs="0"/> <xsd:element name="phone" type="publishable" maxOccurs="1" minOccurs="0"/> </xsd:all> </xsd:complexType> </pre>

Complex Type publishable

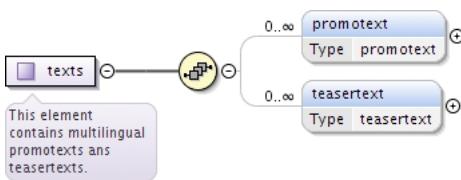
Namespace	No namespace										
Diagram	 <pre> classDiagram publishable < -- xsd:string xsd:string --> Note: Built-in primitive type. The string datatype represents character strings in XML. publishable --> Note: @ publishable publishable --> Note: Type xsd:boolean </pre>										
Type	extension of xsd:string										
Used by	Elements www/facebook, www/homepage, www/myspace, www/phone, www/twitter										
Attributes	<table border="1"> <thead> <tr> <th>QName</th> <th>Type</th> <th>Fixed</th> <th>Default</th> <th>Use</th> </tr> </thead> <tbody> <tr> <td>publishable</td> <td>xsd:boolean</td> <td></td> <td></td> <td>optional</td> </tr> </tbody> </table>	QName	Type	Fixed	Default	Use	publishable	xsd:boolean			optional
QName	Type	Fixed	Default	Use							
publishable	xsd:boolean			optional							
Source	<pre> <xsd:complexType name="publishable"> <xsd:simpleContent> <xsd:extension base="xsd:string"> <xsd:attribute name="publishable" type="xsd:boolean"/> </xsd:extension> </xsd:simpleContent> </xsd:complexType> </pre>										

Complex Type information

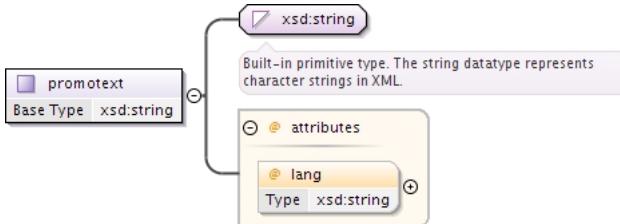
Namespace	No namespace
Annotations	<p>This element contains important data for an item/file. Multilingual promotexts ans teasertexts, dates of physical and digital release, playlength of file, position of the file in relation to other file of bundle, number of set (e.g. 2 for cd 2), the suggested prelistining offset if the file not starts e.g. with significant content, origin country and main language of file and information about related bundles.</p>
Diagram	<pre> classDiagram class information { <<This element contains important data for an item/file. Multilingual promotexts ans teasertexts, dates of physical and...>> } class texts class physical_release_datetime class digital_release_datetime class playlength class num class setnum class suggested_prelistening_offset class origin_country class main_language class related information < -- texts information < -- physical_release_datetime information < -- digital_release_datetime information < -- playlength information < -- num information < -- setnum information < -- suggested_prelistening_offset information < -- origin_country information < -- main_language information < -- related </pre>
Used by	Elements bundle/information, item/information
Model	ALL(texts{0,1} physical_release_datetime digital_release_datetime playlength{0,1} num{0,1} setnum{0,1} suggested_prelistening_offset{0,1} origin_country{0,1} main_language{0,1} related{0,1})
Children	digital_release_datetime, main_language, num, origin_country, physical_release_datetime, playlength, related, setnum, suggested_prelistening_offset, texts
Source	<pre> <xsd:complexType name="information"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains important data for an item/file. Multilingual promotexts ans teasertexts, dates of physical and digital release, playlength of file, position of the file in relation to other file of bundle, number of set (e.g. 2 for cd 2), the suggested prelistining offset if the file not starts e.g. with significant content, origin country and main language of file and information about related bundles.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="texts" type="texts" maxOccurs="1" minOccurs="0"/> <xsd:element name="physical_release_datetime" type="datetimeGMT"/> <xsd:element name="digital_release_datetime" type="datetimeGMT"/> <xsd:element name="playlength" type="xsd:integer" maxOccurs="1" minOccurs="0"/> <xsd:element name="num" type="xsd:integer" maxOccurs="1" minOccurs="0"/> <xsd:element name="setnum" type="xsd:integer" maxOccurs="1" minOccurs="0"/> <xsd:element name="suggested_prelistening_offset" type="xsd:integer" maxOccurs="1" minOccurs="0"/> <xsd:element name="origin_country" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="main_language" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="related" type="related" maxOccurs="1" minOccurs="0"/> </xsd:all> </xsd:complexType> </pre>

Complex Type texts

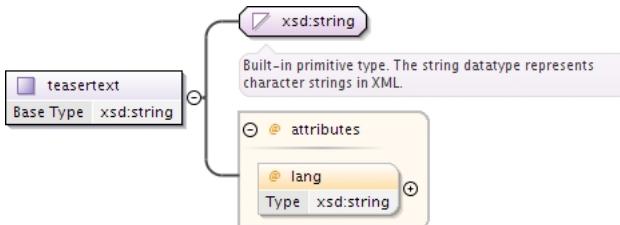
Namespace	No namespace
Annotations	This element contains multilingual promotexts ans teasertexts.

Diagram	
Used by	Element information/texts
Model	promotext*, teasertext*
Children	promotext, teasertext
Source	<pre><xsd:complexType name="texts"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains multilingual promotexts and teasertexts.</xsd:documentation> </xsd:annotation> <xsd:sequence> <xsd:element name="promotext" type="promotext" maxOccurs="unbounded" minOccurs="0"/> <xsd:element name="teasertext" type="teasertext" maxOccurs="unbounded" minOccurs="0"/> </xsd:sequence> </xsd:complexType></pre>

Complex Type promotext

Namespace	No namespace										
Diagram											
Type	extension of xsd:string										
Used by	Element texts/promotext										
Attributes	<table border="1"> <thead> <tr> <th>QName</th> <th>Type</th> <th>Fixed</th> <th>Default</th> <th>Use</th> </tr> </thead> <tbody> <tr> <td>lang</td> <td>xsd:string</td> <td></td> <td></td> <td>optional</td> </tr> </tbody> </table>	QName	Type	Fixed	Default	Use	lang	xsd:string			optional
QName	Type	Fixed	Default	Use							
lang	xsd:string			optional							
Source	<pre><xsd:complexType name="promotext"> <xsd:simpleContent> <xsd:extension base="xsd:string"> <xsd:attribute name="lang" type="xsd:string" /> </xsd:extension> </xsd:simpleContent> </xsd:complexType></pre>										

Complex Type teasertext

Namespace	No namespace										
Diagram											
Type	extension of xsd:string										
Used by	Element texts/teasertext										
Attributes	<table border="1"> <thead> <tr> <th>QName</th> <th>Type</th> <th>Fixed</th> <th>Default</th> <th>Use</th> </tr> </thead> <tbody> <tr> <td>lang</td> <td>xsd:string</td> <td></td> <td></td> <td>optional</td> </tr> </tbody> </table>	QName	Type	Fixed	Default	Use	lang	xsd:string			optional
QName	Type	Fixed	Default	Use							
lang	xsd:string			optional							
Source	<pre><xsd:complexType name="teasertext"></pre>										

```
<xsd:simpleContent>
  <xsd:extension base="xsd:string">
    <xsd:attribute name="lang" type="xsd:string"/>
  </xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
```

Complex Type related

Namespace	No namespace
Annotations	This element contains informations of bundles which are related to the bundle of the actual feed.
Diagram	<p>This element contains informations of bundles which are related to the bundle of the actual feed.</p>
Used by	Element information/related
Model	physical_distributor*, utube{0,1}, bundle*
Children	bundle, physical_distributor, utube
Source	<pre><xsd:complexType name="related"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains informations of bundles which are related to the bundle of the actual feed.</xsd:documentation> </xsd:annotation> <xsd:sequence> <xsd:element name="physical_distributor" type="physical_distributor" maxOccurs="unbounded" minOccurs="0"/> <xsd:element name="utube" type="utube" maxOccurs="1" minOccurs="0"/> <xsd:element name="bundle" type="bundle" maxOccurs="unbounded" minOccurs="0"/> </xsd:sequence> </xsd:complexType></pre>

Complex Type physical_distributor

Namespace	No namespace										
Diagram	<p>Built-in primitive type. The string datatype represents character strings in XML.</p>										
Type	extension of xsd:string										
Used by	Element related/physical_distributor										
Attributes	<table border="1"> <thead> <tr> <th>QName</th> <th>Type</th> <th>Fixed</th> <th>Default</th> <th>Use</th> </tr> </thead> <tbody> <tr> <td>publishable</td> <td>xsd:boolean</td> <td></td> <td></td> <td>optional</td> </tr> </tbody> </table>	QName	Type	Fixed	Default	Use	publishable	xsd:boolean			optional
QName	Type	Fixed	Default	Use							
publishable	xsd:boolean			optional							
Source	<pre><xsd:complexType name="physical_distributor"> <xsd:simpleContent> <xsd:extension base="xsd:string"> <xsd:attribute name="publishable" type="xsd:boolean"/> </xsd:extension> </xsd:simpleContent> </xsd:complexType></pre>										

Complex Type utube

Namespace	No namespace
Annotations	Contains information about youtube url und channel.

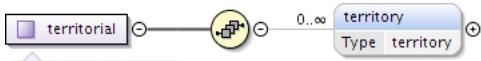
Diagram	
Used by	Element related/utube
Model	ALL(url{0,1} channel{0,1})
Children	channel, url
Source	<pre><xsd:complexType name="utube"> <xsd:annotation> <xsd:documentation xml:lang="en">Contains information about youtube url und channel.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="url" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="channel" type="xsd:string" maxOccurs="1" minOccurs="0"/> </xsd:all> </xsd:complexType></pre>

Complex Type license_basis

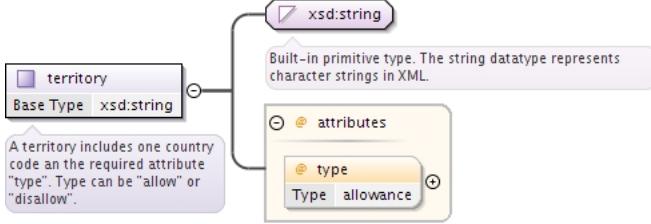
Namespace	No namespace
Annotations	This element includes the basic rules and information under which this bundle is provided.
Diagram	
Used by	Elements bundle/license_basis, item/license_basis
Model	ALL(territorial{0,1} timeframe{0,1} pricing{0,1} streaming_allowed{0,1} channels{0,1} as_on_bundle{0,1})
Children	as_on_bundle, channels, pricing, streaming_allowed, territorial, timeframe
Source	<pre><xsd:complexType name="license_basis"> <xsd:annotation> <xsd:documentation xml:lang="en">This element includes the basic rules and information under which this bundle is provided.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="territorial" type="territorial" maxOccurs="1" minOccurs="0"/> <xsd:element name="timeframe" type="timeframe" maxOccurs="1" minOccurs="0"/> <xsd:element name="pricing" type="pricing" maxOccurs="1" minOccurs="0"/> <xsd:element name="streaming_allowed" type="xsd:boolean" maxOccurs="1" minOccurs="0"/> <xsd:element name="channels" type="channels" maxOccurs="1" minOccurs="0"/> <xsd:element name="as_on_bundle" type="xsd:boolean" maxOccurs="1" minOccurs="0"/> </xsd:all> </xsd:complexType></pre>

Complex Type territorial

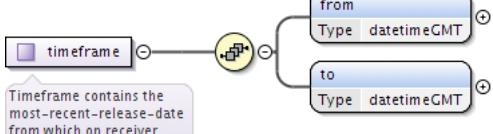
Namespace	No namespace
Annotations	This Element is a container for territories. There should be a entry for all territories with a attribute if distribution is allowed or not.

Diagram	
	This Element is a container for territories. There should be a entry for all territories with a attribute if...
Used by	Element license_basis/territorial
Model	territory*
Children	territory
Source	<pre><xsd:complexType name="territorial"> <xsd:annotation> <xsd:documentation xml:lang="en">This Element is a container for territories. There should be a entry for all territories with a attribute if distribution is allowed or not.</xsd:documentation> </xsd:annotation> <xsd:sequence> <xsd:element name="territory" type="territory" maxOccurs="unbounded" minOccurs="0" /> </xsd:sequence> </xsd:complexType></pre>

Complex Type territory

Namespace	No namespace										
Annotations	A territory includes one country code an the required attribute "type". Type can be "allow" or "disallow".										
Diagram											
Type	extension of xsd:string										
Used by	Element territorial/territory										
Attributes	<table border="1"> <thead> <tr> <th>QName</th> <th>Type</th> <th>Fixed</th> <th>Default</th> <th>Use</th> </tr> </thead> <tbody> <tr> <td>type</td> <td>allowance</td> <td></td> <td></td> <td>optional</td> </tr> </tbody> </table>	QName	Type	Fixed	Default	Use	type	allowance			optional
QName	Type	Fixed	Default	Use							
type	allowance			optional							
Source	<pre><xsd:complexType name="territory"> <xsd:annotation> <xsd:documentation xml:lang="en">A territory includes one country code an the required attribute "type". Type can be "allow" or "disallow".</xsd:documentation> </xsd:annotation> <xsd:simpleContent> <xsd:extension base="xsd:string"> <xsd:attribute name="type" type="allowance" use="optional" /> </xsd:extension> </xsd:simpleContent> </xsd:complexType></pre>										

Complex Type timeframe

Namespace	No namespace
Annotations	Timeframe contains the most-recent-release-date from which on receiver may use this and the cancellation-date.
Diagram	
Used by	Element license_basis/timeframe

Model	from , to
Children	from, to
Source	<pre><xsd:complexType name="timeframe"> <xsd:annotation> <xsd:documentation xml:lang="en">Timeframe contains the most-recent-release-date from which on receiver may use this and the cancellation-date.</xsd:documentation> </xsd:annotation> <xsd:sequence> <xsd:element name="from" type="datetimeGMT"/> <xsd:element name="to" type="datetimeGMT"/> </xsd:sequence> </xsd:complexType></pre>

Complex Type pricing

Namespace	No namespace
Annotations	Pricecode is an arbitrary-info. An explicitly given wholesale-price overrides the basic pricecode-given-wp. Most probably either one of pricecode OR wholesaleprice should be given.
Diagram	<pre> classDiagram class pricing { <<Pricecode is an arbitrary-info. An explicitly given wholesale-price overrides the basic pricecode-given-wp. Most...>> +pricecode +wholesale } class pricecode { <<Type xsd:string>> } class wholesale { <<Type xsd:string>> } pricing < -- pricecode pricing < -- wholesale </pre>
Used by	Element license_basis/pricing
Model	ALL(pricecode{0,1} wholesale{0,1})
Children	pricecode, wholesale
Source	<pre><xsd:complexType name="pricing"> <xsd:annotation> <xsd:documentation xml:lang="en">Pricecode is an arbitrary-info. An explicitly given wholesale-price overrides the basic pricecode-given-wp. Most probably either one of pricecode OR wholesaleprice should be given.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="pricecode" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="wholesale" type="xsd:string" maxOccurs="1" minOccurs="0"/> </xsd:all> </xsd:complexType></pre>

Complex Type channels

Namespace	No namespace
Annotations	This element is a container for channels which can be either "all", "ad supported" or "premium".
Diagram	<pre> classDiagram class channels { <<This element is a container for channels which can be either "all", "ad supported" or "premium".>> +channel } class channel { <<Type channel>> } channels < -- channel </pre>
Used by	Element license_basis/channels
Model	channel*
Children	channel
Source	<pre><xsd:complexType name="channels"> <xsd:annotation> <xsd:documentation xml:lang="en">This element is a container for channels which can be either "all", "ad supported" or "premium".</xsd:documentation> </xsd:annotation> <xsd:sequence> <xsd:element name="channel" type="channel" maxOccurs="unbounded" minOccurs="0"/> </xsd:sequence> </xsd:complexType></pre>

Complex Type channel

Namespace	No namespace										
Annotations	A channels can be either "all", "ad supported" or "premium". The required attribute "type" regards to the allowance in reference to the channel. Type can be "allow" or "disallow".										
Diagram	<pre> classDiagram class channel { <<Base Type xsd:string>> } xsd:string < -- channel channel < -- @type : allowance note over xsd:string: Built-in primitive type. The string datatype represents character strings in XML. note over @type: @ type note over allowance: Type allowance </pre>										
Type	extension of xsd:string										
Used by	Element channels/channel										
Attributes	<table border="1"> <thead> <tr> <th>QName</th><th>Type</th><th>Fixed</th><th>Default</th><th>Use</th></tr> </thead> <tbody> <tr> <td>type</td><td>allowance</td><td></td><td></td><td>required</td></tr> </tbody> </table>	QName	Type	Fixed	Default	Use	type	allowance			required
QName	Type	Fixed	Default	Use							
type	allowance			required							
Source	<pre> <xsd:complexType name="channel"> <xsd:annotation> <xsd:documentation xml:lang="en">A channels can be either "all", "ad supported" or "premium". The required attribute "type" regards to the allowance in reference to the channel. Type can be "allow" or "disallow".</xsd:documentation> </xsd:annotation> <xsd:simpleContent> <xsd:extension base="xsd:string"> <xsd:attribute name="type" type="allowance" use="required"/> </xsd:extension> </xsd:simpleContent> </xsd:complexType> </pre>										

Complex Type license_specifics

Namespace	No namespace
Annotations	This element includes specific rules which should be applied. This can be achieved by given rules.
Diagram	<pre> classDiagram class license_specifics class rules license_specifics --> rules note over license_specifics: This element includes specific rules which should be applied. This can be achieved by given rules. note over rules: rules </pre>
Used by	Elements bundle/license_specifics, item/license_specifics
Model	rules
Children	rules
Source	<pre> <xsd:complexType name="license_specifics"> <xsd:annotation> <xsd:documentation xml:lang="en">This element includes specific rules which should be applied. This can be achieved by given rules.</xsd:documentation> </xsd:annotation> <xsd:sequence> <xsd:element name="rules" type="rules"/> </xsd:sequence> </xsd:complexType> </pre>

Complex Type rules

Namespace	No namespace
Annotations	This element is a container for rules. It needs an ordered mode here - first come first match.
Diagram	<pre> classDiagram class rules class rule rules --> rule note over rules: This element is a container for rules. It needs an ordered mode here - first come first match. note over rule: rule </pre>

Used by	Element	license_specifics/rules
Model	rule*	
Children	rule	
Source		<pre><xsd:complexType name="rules"> <xsd:annotation> <xsd:documentation xml:lang="en">This element is a container for rules. It needs an ordered mode here - first come first match.</xsd:documentation> </xsd:annotation> <xsd:sequence> <xsd:element name="rule" type="rule" maxOccurs="unbounded" minOccurs="0" /> </xsd:sequence> </xsd:complexType></pre>

Complex Type rule

Namespace	No namespace														
Annotations	A rule must include a "if"-element and a "then"-element to shape a legal instruction. It can also include a "else"-element.														
Diagram	<pre> classDiagram class rule { attribute num : xsd:integer element if element then element else } rule < -- rule </pre>														
Used by	Element														
Model	if , then , else{0,1}														
Children	else, if, then														
Attributes	<table border="1"> <thead> <tr> <th>QName</th> <th>Type</th> <th>Fixed</th> <th>Default</th> <th>Use</th> </tr> </thead> <tbody> <tr> <td>num</td> <td>xsd:integer</td> <td></td> <td></td> <td>optional</td> </tr> </tbody> </table>					QName	Type	Fixed	Default	Use	num	xsd:integer			optional
QName	Type	Fixed	Default	Use											
num	xsd:integer			optional											
Source	<pre><xsd:complexType name="rule"> <xsd:annotation> <xsd:documentation xml:lang="en">A rule must include a "if"-element and a "then"-element to shape a legal instruction. It can also include a "else"-element.</xsd:documentation> </xsd:annotation> <xsd:sequence> <xsd:element name="if" type="if"/> <xsd:element name="then" type="then"/> <xsd:element name="else" type="else" maxOccurs="1" minOccurs="0" /> </xsd:sequence> <xsd:attribute name="num" type="xsd:integer"/> </xsd:complexType></pre>														

Complex Type if

Namespace	No namespace				
Annotations	This element must be the first element in a rule. It includes the information what is affected by the rule, an operator like "equals", "before", "after", "contains" or "containedin" and a value which will be compared.				
Diagram	<pre> classDiagram class if { element what : xsd:string element operator : operator element value : xsd:string } if < -- if </pre>				

Used by	Element rule/if
Model	what , operator , value
Children	operator, value, what
Source	<pre><xsd:complexType name="if"> <xsd:annotation> <xsd:documentation xml:lang="en">This element must be the first element in a rule. It includes the information what is affected by the rule, an operator like "equals", "before", "after", "contains" or "containedin" and a value which will be compared.</xsd:documentation> </xsd:annotation> <xsd:sequence> <xsd:element name="what" type="xsd:string"/> <xsd:element name="operator" type="operator"/> <xsd:element name="value" type="xsd:string"/> </xsd:sequence> </xsd:complexType></pre>

Complex Type then

Namespace	No namespace
Annotations	This element must be the second in a rule and includes information "echo" for debugging output and can include an element "break" which means to not process any more rules.
Diagram	
Used by	Element rule/then
Model	echo{0,1} , break{0,1}
Children	break, echo
Source	<pre><xsd:complexType name="then"> <xsd:annotation> <xsd:documentation xml:lang="en">This element must be the second in a rule and includes information "echo" for debugging output and can include an element "break" which means to not process any more rules.</xsd:documentation> </xsd:annotation> <xsd:sequence> <xsd:element name="echo" type="xsd:string" maxOccurs="1" minOccurs="0" /> <xsd:element name="break" maxOccurs="1" minOccurs="0" /> </xsd:sequence> </xsd:complexType></pre>

Complex Type else

Namespace	No namespace
Annotations	This element is optional. It includes information "proclaim" and can include an element "break" which means to not process any more rules.
Diagram	
Used by	Element rule/else
Model	proclaim*, break{0,1}
Children	break, proclaim
Source	<pre><xsd:complexType name="else"></pre>

```

<xsd:annotation>
  <xsd:documentation xml:lang="en">This element is optional. It includes information
  "proclaim" and can include an element "break" which means to not process any more
  rules.</xsd:documentation>
</xsd:annotation>
<xsd:sequence>
  <xsd:element name="proclaim" type="proclaim" maxOccurs="unbounded" minOccurs="0" />
  <xsd:element name="break" maxOccurs="1" minOccurs="0" />
</xsd:sequence>
</xsd:complexType>

```

Complex Type proclaim

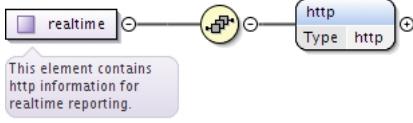
Namespace	No namespace
Annotations	This element includes the information what is affected and the corresponding value.
Diagram	<p>This element includes the information what is affected and the corresponding value.</p>
Used by	Element else/proclaim
Model	what , for
Children	for, what
Source	<pre> <xsd:complexType name="proclaim"> <xsd:annotation> <xsd:documentation xml:lang="en">This element includes the information what is affected and the corresponding value.</xsd:documentation> </xsd:annotation> <xsd:sequence> <xsd:element name="what" type="xsd:string"/> <xsd:element name="for" type="xsd:string"/> </xsd:sequence> </xsd:complexType> </pre>

Complex Type reporting

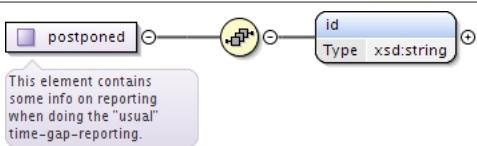
Namespace	No namespace
Annotations	This element contains information about reporting.
Diagram	<p>This element contains information about reporting.</p>
Used by	Elements bundle/reporting, item/reporting
Model	ALL(realtime postponed)
Children	postponed, realtime
Source	<pre> <xsd:complexType name="reporting"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information about reporting.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="realtime" type="realtime"/> <xsd:element name="postponed" type="postponed"/> </xsd:all> </xsd:complexType> </pre>

Complex Type realtime

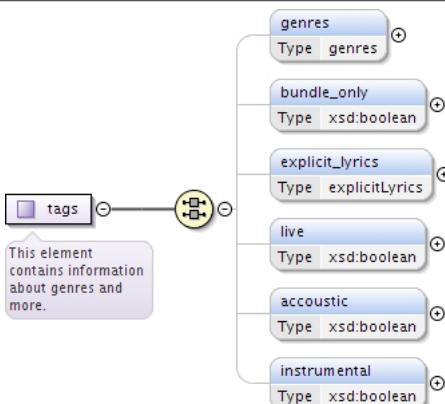
Namespace	No namespace
Annotations	This element contains http information for realtime reporting.

Diagram	 This element contains http information for realtime reporting.
Used by	Element reporting/realtime
Model	http
Children	http
Source	<pre><xsd:complexType name="realtime"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains http information for realtime reporting.</xsd:documentation> </xsd:annotation> <xsd:sequence> <xsd:element name="http" type="http" /> </xsd:sequence> </xsd:complexType></pre>

Complex Type postponed

Namespace	No namespace
Annotations	This element contains some info on reporting when doing the "usual" time-gap-reporting.
Diagram	 This element contains some info on reporting when doing the "usual" time-gap-reporting.
Used by	Element reporting/postponed
Model	id
Children	id
Source	<pre><xsd:complexType name="postponed"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains some info on reporting when doing the "usual" time-gap-reporting.</xsd:documentation> </xsd:annotation> <xsd:sequence> <xsd:element name="id" type="xsd:string" /> </xsd:sequence> </xsd:complexType></pre>

Complex Type tags

Namespace	No namespace
Annotations	This element contains information about genres and more.
Diagram	 This element contains information about genres and more.
Used by	Elements bundle/tags, item/tags
Model	ALL(genres{0,1} bundle_only{0,1} explicit_lyrics{0,1} live{0,1} acoustic{0,1} instrumental{0,1})

Children	accoustic, bundle_only, explicit_lyrics, genres, instrumental, live
Source	<pre><xsd:complexType name="tags"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information about genres and more.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="genres" type="genres" maxOccurs="1" minOccurs="0"/> <xsd:element name="bundle_only" type="xsd:boolean" maxOccurs="1" minOccurs="0"/> <xsd:element name="explicit_lyrics" type="explicitLyrics" maxOccurs="1" minOccurs="0"/> </xsd:all> </xsd:complexType></pre>

Complex Type genres

Namespace	No namespace
Annotations	This element contains a list of genres.
Diagram	<pre>graph LR genres[genres] -- "0..∞" --> genre[genre] subgraph genres direction TB genres --- genre end genre --- type[xsd:string] genre --- xsdString[xsd:string]</pre>
Used by	Element tags/genres
Model	genre*
Children	genre
Source	<pre><xsd:complexType name="genres"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains a list of genres.</xsd:documentation> </xsd:annotation> <xsd:sequence> <xsd:element name="genre" type="xsd:string" maxOccurs="unbounded" minOccurs="0"/> </xsd:sequence> </xsd:complexType></pre>

Complex Type files

Namespace	No namespace
Annotations	This element contains a list of files.
Diagram	<pre>graph LR files[files] -- "0..∞" --> file[file] subgraph files direction TB files --- file end file --- type[file] file --- xsdType[xsd:type]</pre>
Used by	Elements bundle/files, item/files
Model	file*
Children	file
Source	<pre><xsd:complexType name="files"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains a list of files.</xsd:documentation> </xsd:annotation> <xsd:sequence> <xsd:element name="file" type="file" maxOccurs="unbounded" minOccurs="0"/> </xsd:sequence> </xsd:complexType></pre>

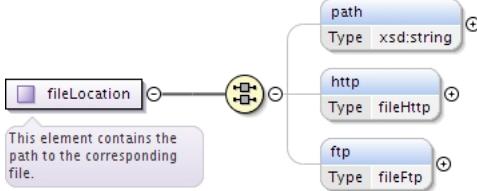
Complex Type file

Namespace	No namespace
Annotations	This element contains information and location of a file.

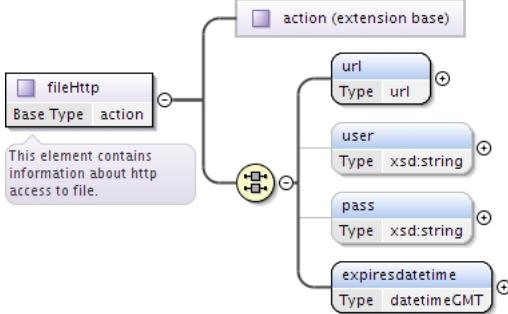
Diagram	<pre> classDiagram class file { <<This element contains information and location of a file.>> } class fileLocation { location "fileLocation" type "fileType" filetype "xsd:string" samplerate "xsd:string" samplesize "xsd:string" bitrate "xsd:string" bitratetype "xsd:string" codec "xsd:string" codecsettings "xsd:string" bytes "xsd:integer" checksums "checksums" channels "fileChannels" dimension "dimension" decryptinfo "decryptinfo" } file "0..1" --> "1..1" fileLocation </pre>
Used by	Element files/file
Model	ALL(location type{0,1} filetype{0,1} samplerate{0,1} samplesize{0,1} bitrate{0,1} bitratetype{0,1} codec{0,1} codecsettings{0,1} bytes{0,1} checksums channels{0,1} dimension{0,1} decryptinfo{0,1})
Children	bitrate, bitratetype, bytes, channels, checksums, codec, codecsettings, decryptinfo, dimension, filetype, location, samplerate, samplesize, type
Source	<pre> <xsd:complexType name="file"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information and location of a file.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="location" type="fileLocation"/> <xsd:element name="type" type="fileType" maxOccurs="1" minOccurs="0"/> <xsd:element name="filetype" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="samplerate" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="samplesize" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="bitrate" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="bitratetype" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="codec" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="codecsettings" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="bytes" type="xsd:integer" maxOccurs="1" minOccurs="0"/> <xsd:element name="checksums" type="checksums"/> <xsd:element name="channels" type="fileChannels" maxOccurs="1" minOccurs="0"/> <xsd:element name="dimension" type="dimension" maxOccurs="1" minOccurs="0"/> <xsd:element name="decryptinfo" type="decryptinfo" maxOccurs="1" minOccurs="0"/> </xsd:all> </xsd:complexType> </pre>

Complex Type fileLocation

Namespace	No namespace
Annotations	This element contains the path to the corresponding file.

Diagram	
Used by	Element file/location
Model	ALL(path{0,1} http{0,1} ftp{0,1})
Children	ftp, http, path
Source	<pre><xsd:complexType name="fileLocation"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains the path to the corresponding file.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="path" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="http" type="fileHttp" maxOccurs="1" minOccurs="0"/> <xsd:element name="ftp" type="fileFtp" maxOccurs="1" minOccurs="0"/> </xsd:all> </xsd:complexType></pre>

Complex Type fileHttp

Namespace	No namespace
Annotations	This element contains information about http access to file.
Diagram	
Type	extension of action
Type hierarchy	<ul style="list-style-type: none"> • action • fileHttp
Used by	Element fileLocation/http
Model	ALL(url user{0,1} pass{0,1} expiresdatetime)
Children	expiresdatetime, pass, url, user
Source	<pre><xsd:complexType name="fileHttp"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information about http access to file.</xsd:documentation> </xsd:annotation> <xsd:complexContent> <xsd:extension base="action"> <xsd:all> <xsd:element name="url" type="url"/> <xsd:element name="user" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="pass" type="xsd:string" maxOccurs="1" minOccurs="0"/> <xsd:element name="expiresdatetime" type="datetimeGMT"/> </xsd:all> </xsd:extension> </xsd:complexContent> </xsd:complexType></pre>

Complex Type fileFtp

Namespace	No namespace
Annotations	This element contains information about ftp access to file.

Diagram	<pre> classDiagram class fileFtp { <<Base Type: action>> <<This element contains information about ftp access to file.>> } class action { <<extension base>> } fileFtp < -- action action < --> server action < --> port action < --> path action < --> user action < --> pass action < --> expiresdatetime </pre>
Type	extension of action
Type hierarchy	<ul style="list-style-type: none"> • action • fileFtp
Used by	Element fileLocation/ftp
Model	ALL(server port path user{0,1} pass{0,1} expiresdatetime)
Children	expiresdatetime, pass, path, port, server, user
Source	<pre> <xsd:complexType name="fileFtp"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information about ftp access to file.</xsd:documentation> </xsd:annotation> <xsd:complexContent> <xsd:extension base="action"> <xsd:all> <xsd:element name="server" type="xsd:string" /> <xsd:element name="port" type="xsd:string" /> <xsd:element name="path" type="xsd:string" /> <xsd:element name="user" type="xsd:string" maxOccurs="1" minOccurs="0" /> <xsd:element name="pass" type="xsd:string" maxOccurs="1" minOccurs="0" /> <xsd:element name="expiresdatetime" type="datetimeGMT" /> </xsd:all> </xsd:extension> </xsd:complexContent> </xsd:complexType> </pre>

Complex Type checksums

Namespace	No namespace
Annotations	This element contains checksums for the file.
Diagram	<pre> classDiagram class checksums { <<This element contains checksums for the file.>> } class file { <<extension base>> } checksums < -- file file < --> md5 file < --> sha1 file < --> sha256 </pre>
Used by	Elements decryptinfo/checksums, file/checksums
Model	ALL(md5{0,1} sha1{0,1} sha256{0,1})
Children	md5, sha1, sha256
Source	<pre> <xsd:complexType name="checksums"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains checksums for the file.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="md5" type="xsd:string" maxOccurs="1" minOccurs="0" /> <xsd:element name="sha1" type="xsd:string" maxOccurs="1" minOccurs="0" /> <xsd:element name="sha256" type="xsd:string" maxOccurs="1" minOccurs="0" /> </xsd:all> </xsd:complexType> </pre>

</xsd:complexType>

Complex Type dimension

Namespace	No namespace
Annotations	This element contains entries for width and height of the file.
Diagram	<pre> classDiagram class dimension { <<This element contains entries for width and height of the file.>> width height } width { Type xsd:integer } height { Type xsd:integer } </pre>
Used by	Element file/dimension
Model	width , height
Children	height, width
Source	<pre> <xsd:complexType name="dimension"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains entries for width and height of the file.</xsd:documentation> </xsd:annotation> <xsd:sequence> <xsd:element name="width" type="xsd:integer" /> <xsd:element name="height" type="xsd:integer" /> </xsd:sequence> </xsd:complexType> </pre>

Complex Type decryptinfo

Namespace	No namespace
Annotations	This element contains information about decryption of corresponding file.
Diagram	<pre> classDiagram class decryptinfo { <<This element contains information about decryption of corresponding file.>> cipher initvector key bytes checksums } cipher { Type xsd:string } initvector { Type xsd:string } key { Type xsd:string } bytes { Type xsd:string } checksums { Type checksums } </pre>
Used by	Element file/decryptinfo
Model	ALL(cipher{0,1} initvector{0,1} key{0,1} bytes{0,1} checksums{0,1})
Children	bytes, checksums, cipher, initvector, key
Source	<pre> <xsd:complexType name="decryptinfo"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information about decryption of corresponding file.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="cipher" type="xsd:string" minOccurs="0" maxOccurs="1"/> <xsd:element name="initvector" type="xsd:string" minOccurs="0" maxOccurs="1"/> <xsd:element name="key" type="xsd:string" minOccurs="0" maxOccurs="1"/> <xsd:element name="bytes" type="xsd:string" minOccurs="0" maxOccurs="1"/> <xsd:element name="checksums" type="checksums" minOccurs="0" maxOccurs="1"/> </xsd:all> </xsd:complexType> </pre>

Complex Type purchase

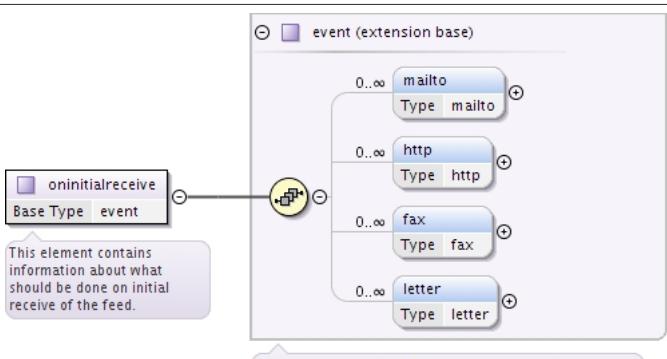
Namespace	No namespace
Annotations	This element contains information about purchase. Mostly when this feeds recipient is a POS.

Diagram	
Used by	Element bundle/purchase
Model	ALL(pos url)
Children	pos, url
Source	<pre><xsd:complexType name="purchase"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information about purchase. Mostly when this feeds recipient is a POS.</xsd:documentation> </xsd:annotation> <xsd:all> <xsd:element name="pos" type="xsd:string"/> <xsd:element name="url" type="xsd:string"/> </xsd:all> </xsd:complexType></pre>

Complex Type `fingerprint`

Namespace	No namespace
Annotations	This element includes an element "echoprint" (http://echoprint.me https://github.com/echonest/echoprint-codegen).
Diagram	http://echoprint.me https://github.com/echonest/echoprint-codegen).'" data-bbox="265 430 565 490"/>
Used by	Element item/fingerprint
Model	echoprint{0,1}
Children	echoprint
Source	<pre><xsd:complexType name="fingerprint"> <xsd:annotation> <xsd:documentation xml:lang="en">This element includes an element "echoprint" (http://echoprint.me https://github.com/echonest/echoprint-codegen).</xsd:documentation> </xsd:annotation> <xsd:sequence> <xsd:element name="echoprint" type="xsd:string" maxOccurs="1" minOccurs="0"/> </xsd:sequence> </xsd:complexType></pre>

Complex Type `oninitialreceive`

Namespace	No namespace
Annotations	This element contains information about what should be done on initial receive of the feed.
Diagram	

Type	extension of event
Type hierarchy	<ul style="list-style-type: none"> • event <ul style="list-style-type: none"> • oninitialreceive
Model	mailto*, http*, fax*, letter*
Children	fax, http, letter, mailto
Source	<pre><xsd:complexType name="oninitialreceive"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information about what should be done on initial receive of the feed.</xsd:documentation> </xsd:annotation> <xsd:complexContent> <xsd:extension base="event"/> </xsd:complexContent> </xsd:complexType></pre>

Complex Type onprocessstart

Namespace	No namespace
Annotations	This element contains information about what should be done on the start of processing the feed.
Diagram	<p>The diagram illustrates the UML representation of the 'onprocessstart' complex type. It shows 'onprocessstart' as an extension of the 'event' base type. The 'onprocessstart' type has a multiplicity of 0..oo and is associated with four children: 'mailto' (Type mailto), 'http' (Type http), 'fax' (Type fax), and 'letter' (Type letter). A callout box provides the annotation text: 'This element contains information about what should be done on the start of processing the feed.'</p>
Type	extension of event
Type hierarchy	<ul style="list-style-type: none"> • event <ul style="list-style-type: none"> • onprocessstart
Model	mailto*, http*, fax*, letter*
Children	fax, http, letter, mailto
Source	<pre><xsd:complexType name="onprocessstart"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information about what should be done on the start of processing the feed.</xsd:documentation> </xsd:annotation> <xsd:complexContent> <xsd:extension base="event"/> </xsd:complexContent> </xsd:complexType></pre>

Complex Type onprocessend

Namespace	No namespace
Annotations	This element contains information about what should be done on the end of processing the feed.

Diagram	<p>This element contains information about what should be done on the end of processing the feed.</p> <p>This element contains information about possible events and actions.</p>
Type	extension of event
Type hierarchy	<ul style="list-style-type: none"> • event • onprocessend
Model	mailto*, http*, fax*, letter*
Children	fax, http, letter, mailto
Source	<pre><xsd:complexType name="onprocessend"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information about what should be done on the end of processing the feed.</xsd:documentation> </xsd:annotation> <xsd:complexContent> <xsd:extension base="event" /> </xsd:complexContent> </xsd:complexType></pre>

Complex Type onfullsuccess

Namespace	No namespace
Annotations	This element contains information about what should be done on full success processing the feed.
Diagram	<p>This element contains information about what should be done on full success processing the feed.</p> <p>This element contains information about possible events and actions.</p>
Type	extension of event
Type hierarchy	<ul style="list-style-type: none"> • event • onfullsuccess
Model	mailto*, http*, fax*, letter*
Children	fax, http, letter, mailto
Source	<pre><xsd:complexType name="onfullsuccess"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information about what should be done on full success processing the feed.</xsd:documentation> </xsd:annotation> <xsd:complexContent> <xsd:extension base="event" /> </xsd:complexContent> </xsd:complexType></pre>

Complex Type onerror

Namespace	No namespace
Annotations	This element contains information about what should be done on error processing the feed.
Diagram	<pre> classDiagram class onerror { <<Base Type event>> } class event { <<extension base>> } onerror --> event onerror --> mailto onerror --> http onerror --> fax onerror --> letter class mailto class http class fax class letter </pre> <p>This element contains information about what should be done on error processing the feed.</p> <p>This element contains information about possible events and actions.</p>
Type	extension of event
Type hierarchy	<ul style="list-style-type: none"> • event • onerror
Model	mailto*, http*, fax*, letter*
Children	fax, http, letter, mailto
Source	<pre> <xsd:complexType name="onerror"> <xsd:annotation> <xsd:documentation xml:lang="en">This element contains information about what should be done on error processing the feed.</xsd:documentation> </xsd:annotation> <xsd:complexContent> <xsd:extension base="event" /> </xsd:complexContent> </xsd:complexType> </pre>

Simple Type(s)

Simple Type datetimetypeGMT

Namespace	No namespace
Diagram	<pre> classDiagram class datetimetypeGMT { <<Built-in primitive type. The string datatype represents character strings in XML.>> } class xsd:string datetimetypeGMT --> xsd:string </pre> <p>Built-in primitive type. The string datatype represents character strings in XML.</p>
Type	restriction of xsd:string
Facets	<p>pattern</p> $\begin{aligned} &\backslash d\{4\}-\backslash d\{2\}-\backslash d\{2\} \\ &\backslash d\{2\}:\backslash d\{2\}:\backslash d\{2\} \text{ GMT} \\ &+\backslash d\{2\}:\backslash d\{2\} \end{aligned}$
Used by	<p>Elements</p> <p>feedinfo/creationdatetime, feedinfo/effectivedatetime, fileFtp/expiredatetime, fileHttp/expiredatetime, information/digital_release_datetime, information/physical_release_datetime, timeframe/from, timeframe/to</p>
Source	<pre> <xsd:simpleType name="datetimetypeGMT"> <xsd:restriction base="xsd:string"> <xsd:pattern value="\backslash d\{4\}-\backslash d\{2\}-\backslash d\{2\} \backslash d\{2\}:\backslash d\{2\}:\backslash d\{2\} \text{ GMT}\+\backslash d\{2\}:\backslash d\{2\}" /> <!-- "2010-01-31 00:00:00 GMT+00:00" - should be altered to some NMTOKENS or such ... --> </xsd:restriction> </xsd:simpleType> </pre>

Simple Type email

Namespace	No namespace
Diagram	<pre> classDiagram class email { <<Built-in primitive type. The string datatype represents character strings in XML.>> } class xsd:string email --> xsd:string </pre> <p>Built-in primitive type. The string datatype represents character strings in XML.</p>

Type	xsd:string
Used by	Elements creator/email, crypto/relatedemail, licensee/email, licensor/email, sender/email
Source	<pre><xsd:simpleType name="email"> <xsd:restriction base="xsd:string"/> </xsd:simpleType></pre>

Simple Type userid

Namespace	No namespace
Diagram	<pre> graph LR userid[xsd:userid] --> xsdstring[xsd:string] </pre> <p>Built-in primitive type. The string datatype represents character strings in XML.</p>
Type	xsd:string
Used by	Element creator/userid
Source	<pre><xsd:simpleType name="userid"> <xsd:restriction base="xsd:string"/> </xsd:simpleType></pre>

Simple Type receivertypes

Namespace	No namespace								
Diagram	<pre> graph LR receivertypes[xsd:receivertypes] --> xsdstring[xsd:string] </pre> <p>Built-in primitive type. The string datatype represents character strings in XML.</p>								
Type	restriction of xsd:string								
Facets	<table border="1"> <tr> <td>enumeration</td> <td>ftp</td> </tr> <tr> <td>enumeration</td> <td>ftps</td> </tr> <tr> <td>enumeration</td> <td>sftp</td> </tr> <tr> <td>enumeration</td> <td>openSDX-Beam</td> </tr> </table>	enumeration	ftp	enumeration	ftps	enumeration	sftp	enumeration	openSDX-Beam
enumeration	ftp								
enumeration	ftps								
enumeration	sftp								
enumeration	openSDX-Beam								
Used by	Element receiver/type								
Source	<pre><xsd:simpleType name="receivertypes"> <xsd:restriction base="xsd:string"> <xsd:enumeration value="ftp"/> <xsd:enumeration value="ftps"/> <xsd:enumeration value="sftp"/> <xsd:enumeration value="openSDX-Beam"/> </xsd:restriction> </xsd:simpleType></pre>								

Simple Type iporhostname

Namespace	No namespace
Diagram	<pre> graph LR iporhostname[xsd:iporhostname] --> xsdstring[xsd:string] </pre> <p>Built-in primitive type. The string datatype represents character strings in XML.</p>
Type	xsd:string
Used by	Element receiver/servername
Source	<pre><xsd:simpleType name="iporhostname"> <xsd:restriction base="xsd:string"> </xsd:restriction> </xsd:simpleType></pre>

Simple Type ipv4

Namespace	No namespace
Diagram	<pre> graph LR ipv4[xsd:ipv4] --> xsdstring[xsd:string] </pre> <p>Built-in primitive type. The string datatype represents character strings in XML.</p>

Type	restriction of xsd:string	
Facets	pattern	\d{3}\.\d{3}\.\d{3}\.\d{3}
Used by	Element	receiver/serverip4
Source	<pre><xsd:simpleType name="ipv4"> <xsd:restriction base="xsd:string"> <xsd:pattern value="\d{3}\.\d{3}\.\d{3}\.\d{3}" /> <!-- should be checked for 0-255 etc. --> </xsd:restriction> </xsd:simpleType></pre>	

Simple Type ipv6

Namespace	No namespace
Diagram	<p>Built-in primitive type. The string datatype represents character strings in XML.</p>
Type	xsd:string
Used by	Element receiver/serveripv6
Source	<pre><xsd:simpleType name="ipv6"> <xsd:restriction base="xsd:string"> <!-- not pattern defined yet... --> </xsd:restriction> </xsd:simpleType></pre>

Simple Type authtype

Namespace	No namespace										
Diagram	<p>Built-in primitive type. The string datatype represents character strings in XML.</p>										
Type	restriction of xsd:string										
Facets	<table border="0"> <tr> <td>enumeration</td> <td>login</td> </tr> <tr> <td>enumeration</td> <td>keyfile</td> </tr> <tr> <td>enumeration</td> <td>kerberos</td> </tr> <tr> <td>enumeration</td> <td>keyfile+login</td> </tr> <tr> <td>enumeration</td> <td>keyfile+username</td> </tr> </table>	enumeration	login	enumeration	keyfile	enumeration	kerberos	enumeration	keyfile+login	enumeration	keyfile+username
enumeration	login										
enumeration	keyfile										
enumeration	kerberos										
enumeration	keyfile+login										
enumeration	keyfile+username										
Used by	Element receiver/authtype										
Source	<pre><xsd:simpleType name="authtype"> <xsd:restriction base="xsd:string"> <xsd:enumeration value="login"/> <xsd:enumeration value="keyfile"/> <xsd:enumeration value="kerberos"/> <xsd:enumeration value="keyfile+login"/> <xsd:enumeration value="keyfile+username"/> </xsd:restriction> </xsd:simpleType></pre>										

Simple Type keyid

Namespace	No namespace
Diagram	<p>Built-in primitive type. The string datatype represents character strings in XML.</p>
Type	xsd:string
Used by	Element crypto/usedkeyid
Source	<pre><xsd:simpleType name="keyid"> <xsd:restriction base="xsd:string"> </xsd:restriction> </xsd:simpleType></pre>

| </xsd:simpleType> |

Simple Type emaillist

Namespace	No namespace
Diagram	<pre> graph LR emaillist[xsd:emaillist] --> xsdstring[xsd:string] </pre> <p>Built-in primitive type. The string datatype represents character strings in XML.</p>
Type	xsd:string
Used by	Element mailto/receiver
Source	<pre> <xsd:simpleType name="emaillist"> <xsd:restriction base="xsd:string"> <!-- make to NMTOKENS or such... --> </xsd:restriction> </xsd:simpleType> </pre>

Simple Type url

Namespace	No namespace
Diagram	<pre> graph LR url[xsd:url] --> xsdstring[xsd:string] </pre> <p>Built-in primitive type. The string datatype represents character strings in XML.</p>
Type	xsd:string
Used by	Elements fileHttp/url, http/url
Source	<pre> <xsd:simpleType name="url"> <xsd:restriction base="xsd:string"> <!-- not pattern defined yet... --> </xsd:restriction> </xsd:simpleType> </pre>

Simple Type httpmethods

Namespace	No namespace						
Diagram	<pre> graph LR httpmethods[xsd:httpmethods] --> xsdstring[xsd:string] </pre> <p>Built-in primitive type. The string datatype represents character strings in XML.</p>						
Type	restriction of xsd:string						
Facets	<table> <tr> <td>enumeration</td> <td>GET</td> </tr> <tr> <td>enumeration</td> <td>POST</td> </tr> <tr> <td>enumeration</td> <td>HEAD</td> </tr> </table>	enumeration	GET	enumeration	POST	enumeration	HEAD
enumeration	GET						
enumeration	POST						
enumeration	HEAD						
Used by	Element http/type						
Source	<pre> <xsd:simpleType name="httpmethods"> <xsd:restriction base="xsd:string"> <xsd:enumeration value="GET"/> <xsd:enumeration value="POST"/> <xsd:enumeration value="HEAD"/> </xsd:restriction> </xsd:simpleType> </pre>						

Simple Type contributorType

Namespace	No namespace				
Diagram	<pre> graph LR contributorType[xsd:contributorType] --> xsdstring[xsd:string] </pre> <p>Built-in primitive type. The string datatype represents character strings in XML.</p>				
Type	restriction of xsd:string				
Facets	<table> <tr> <td>enumeration</td> <td>label</td> </tr> <tr> <td>enumeration</td> <td>performer</td> </tr> </table>	enumeration	label	enumeration	performer
enumeration	label				
enumeration	performer				

	enumeration	texter
	enumeration	writer
	enumeration	vocals
	enumeration	conductor
	enumeration	display_artist
	enumeration	compilator
	enumeration	copyright
	enumeration	production
	enumeration	clearinghouse
Used by	Element	contributor/type
Source	<pre><xsd:simpleType name="contributorType"> <xsd:restriction base="xsd:string"> <xsd:enumeration value="label"/> <xsd:enumeration value="performer"/> <xsd:enumeration value="texter"/> <xsd:enumeration value="writer"/> <xsd:enumeration value="vocals"/> <xsd:enumeration value="conductor"/> <xsd:enumeration value="display_artist"/> <xsd:enumeration value="compilator"/> <xsd:enumeration value="copyright"/> <xsd:enumeration value="production"/> <xsd:enumeration value="clearinghouse"/> </xsd:restriction> </xsd:simpleType></pre>	

Simple Type allowance

Namespace	No namespace				
Diagram					
Type	restriction of xsd:string				
Facets	<table> <tr> <td>enumeration</td> <td>allow</td> </tr> <tr> <td>enumeration</td> <td>disallow</td> </tr> </table>	enumeration	allow	enumeration	disallow
enumeration	allow				
enumeration	disallow				
Used by	Attributes channel/@type, territory/@type				
Source	<pre><xsd:simpleType name="allowance"> <xsd:restriction base="xsd:string"> <xsd:enumeration value="allow"/> <xsd:enumeration value="disallow"/> </xsd:restriction> </xsd:simpleType></pre>				

Simple Type operator

Namespace	No namespace										
Diagram											
Type	restriction of xsd:string										
Facets	<table> <tr> <td>enumeration</td> <td>equals</td> </tr> <tr> <td>enumeration</td> <td>before</td> </tr> <tr> <td>enumeration</td> <td>after</td> </tr> <tr> <td>enumeration</td> <td>contains</td> </tr> <tr> <td>enumeration</td> <td>containedin</td> </tr> </table>	enumeration	equals	enumeration	before	enumeration	after	enumeration	contains	enumeration	containedin
enumeration	equals										
enumeration	before										
enumeration	after										
enumeration	contains										
enumeration	containedin										
Used by	Element if/operator										
Source	<pre><xsd:simpleType name="operator"> <xsd:restriction base="xsd:string"></pre>										

```

<xsd:enumeration value="equals" />
<xsd:enumeration value="before" />
<xsd:enumeration value="after" />
<xsd:enumeration value="contains" />
<xsd:enumeration value="containedin" />
</xsd:restriction>
</xsd:simpleType>

```

Simple Type explicitLyrics

Namespace	No namespace						
Diagram							
Type	restriction of xsd:string						
Facets	<table> <tr> <td>enumeration</td> <td>true</td> </tr> <tr> <td>enumeration</td> <td>false</td> </tr> <tr> <td>enumeration</td> <td>cleaned</td> </tr> </table>	enumeration	true	enumeration	false	enumeration	cleaned
enumeration	true						
enumeration	false						
enumeration	cleaned						
Used by	Element tags/explicit_lyrics						
Source	<pre> <xsd:simpleType name="explicitLyrics"> <xsd:restriction base="xsd:string"> <xsd:enumeration value="true" /> <xsd:enumeration value="false" /> <xsd:enumeration value="cleaned" /> </xsd:restriction> </xsd:simpleType> </pre>						

Simple Type fileType

Namespace	No namespace								
Diagram									
Type	restriction of xsd:string								
Facets	<table> <tr> <td>enumeration</td> <td>full</td> </tr> <tr> <td>enumeration</td> <td>prelistening</td> </tr> <tr> <td>enumeration</td> <td>cover</td> </tr> <tr> <td>enumeration</td> <td>booklet</td> </tr> </table>	enumeration	full	enumeration	prelistening	enumeration	cover	enumeration	booklet
enumeration	full								
enumeration	prelistening								
enumeration	cover								
enumeration	booklet								
Used by	Element file/type								
Source	<pre> <xsd:simpleType name="fileType"> <xsd:restriction base="xsd:string"> <xsd:enumeration value="full" /> <xsd:enumeration value="prelistening" /> <xsd:enumeration value="cover" /> <xsd:enumeration value="booklet" /> </xsd:restriction> </xsd:simpleType> </pre>								

Simple Type fileChannels

Namespace	No namespace								
Diagram									
Type	restriction of xsd:string								
Facets	<table> <tr> <td>enumeration</td> <td>mono</td> </tr> <tr> <td>enumeration</td> <td>stereo</td> </tr> <tr> <td>enumeration</td> <td>joint-stereo</td> </tr> <tr> <td>enumeration</td> <td>5.1</td> </tr> </table>	enumeration	mono	enumeration	stereo	enumeration	joint-stereo	enumeration	5.1
enumeration	mono								
enumeration	stereo								
enumeration	joint-stereo								
enumeration	5.1								

Used by	Element	file/channels
Source		<pre><xsd:simpleType name="fileChannels"> <xsd:restriction base="xsd:string"> <xsd:enumeration value="mono"/> <xsd:enumeration value="stereo"/> <xsd:enumeration value="joint-stereo"/> <xsd:enumeration value="5.1"/> </xsd:restriction> </xsd:simpleType></pre>

Attribute(s)

Attribute **publishable** / @publishable

Namespace	No namespace	
Type	xsd:boolean	
Properties	content: simple	
Used by	Complex Type	publishable
Source	<pre><xsd:attribute name="publishable" type="xsd:boolean" /></pre>	

Attribute **contributor** / @num

Namespace	No namespace	
Type	xsd:integer	
Properties	content: simple	
Used by	Complex Type	contributor
Source	<pre><xsd:attribute name="num" type="xsd:integer" /></pre>	

Attribute **promotext** / @lang

Namespace	No namespace	
Type	xsd:string	
Properties	content: simple	
Used by	Complex Type	promotext
Source	<pre><xsd:attribute name="lang" type="xsd:string" /></pre>	

Attribute **teaserText** / @lang

Namespace	No namespace	
Type	xsd:string	
Properties	content: simple	
Used by	Complex Type	teaserText
Source	<pre><xsd:attribute name="lang" type="xsd:string" /></pre>	

Attribute **physical_distributor** / @publishable

Namespace	No namespace	
Type	xsd:boolean	
Properties	content: simple	
Used by	Complex Type	physical_distributor
Source	<pre><xsd:attribute name="publishable" type="xsd:boolean" /></pre>	

Attribute **territory** / @type

Namespace	No namespace
-----------	--------------

Type	allowance	
Properties	use: optional	
Facets	enumeration	allow
	enumeration	disallow
Used by	Complex Type	territory
Source	<code><xsd:attribute name="type" type="allowance" use="optional"/></code>	

Attribute channel / @type

Namespace	No namespace	
Type	allowance	
Properties	use:	required
Facets	enumeration	allow
	enumeration	disallow
Used by	Complex Type	channel
Source	<code><xsd:attribute name="type" type="allowance" use="required"/></code>	

Attribute rule / @num

Namespace	No namespace	
Type	xsd:integer	
Properties	content:	simple
Used by	Complex Type	rule
Source	<code><xsd:attribute name="num" type="xsd:integer"/></code>	