

XML Schema Documentation

Table of Contents

- [Schema Document Properties](#)
- [Global Definitions](#)
 - [Complex Type: aeg](#)
 - [Complex Type: automated-parameter](#)
 - [Complex Type: automation-control](#)
 - [Complex Type: automation-mapping-base](#)
 - [Complex Type: axis](#)
 - [Complex Type: bank](#)
 - [Complex Type: bounds](#)
 - [Complex Type: curve](#)
 - [Complex Type: env](#)
 - [Complex Type: filter](#)
 - [Complex Type: fx](#)
 - [Complex Type: group](#)
 - [Complex Type: id-ref](#)
 - [Complex Type: lfo](#)
 - [Complex Type: lfobase](#)
 - [Complex Type: loop](#)
 - [Complex Type: mapping](#)
 - [Complex Type: midi-control](#)
 - [Complex Type: modulation](#)
 - [Complex Type: param](#)
 - [Complex Type: performance](#)
 - [Complex Type: program](#)
 - [Complex Type: region](#)
 - [Complex Type: send](#)
 - [Complex Type: seq](#)
 - [Complex Type: slot](#)
 - [Complex Type: soundshape](#)
 - [Complex Type: soundshape-base](#)
 - [Complex Type: split](#)
 - [Complex Type: versioned-element](#)
 - [Complex Type: wave](#)
 - [Complex Type: wavematrix](#)
 - [Model Group: sound-elements](#)
 - [Simple Type: automation](#)
 - [Simple Type: automation-pin](#)
 - [Simple Type: bpm](#)
 - [Simple Type: channel](#)
 - [Simple Type: chokeGroup](#)
 - [Simple Type: coarse](#)
 - [Simple Type: curve-shape](#)
 - [Simple Type: filterslope](#)
 - [Simple Type: filtertype](#)
 - [Simple Type: fine](#)
 - [Simple Type: freq](#)
 - [Simple Type: fx-bus](#)
 - [Simple Type: glide-mode](#)
 - [Simple Type: group-output](#)
 - [Simple Type: interpolation](#)
 - [Simple Type: lfomode](#)
 - [Simple Type: lftype](#)
 - [Simple Type: loopmode](#)
 - [Simple Type: midicc](#)

- [Simple Type: **midinote**](#)
- [Simple Type: **midivalue**](#)
- [Simple Type: **moddst**](#)
- [Simple Type: **modsrc**](#)
- [Simple Type: **mute**](#)
- [Simple Type: **noteprio**](#)
- [Simple Type: **output**](#)
- [Simple Type: **pan**](#)
- [Simple Type: **percent**](#)
- [Simple Type: **playback**](#)
- [Simple Type: **playmode**](#)
- [Simple Type: **pluginversion**](#)
- [Simple Type: **polar-percent**](#)
- [Simple Type: **poly**](#)
- [Simple Type: **polymode**](#)
- [Simple Type: **program-change-mode**](#)
- [Simple Type: **quality**](#)
- [Simple Type: **rootkey**](#)
- [Simple Type: **round-robin-source**](#)
- [Simple Type: **scale**](#)
- [Simple Type: **send-destination**](#)
- [Simple Type: **seqlength**](#)
- [Simple Type: **sequence**](#)
- [Simple Type: **solo**](#)
- [Simple Type: **synctype**](#)
- [Simple Type: **tempo**](#)
- [Simple Type: **time**](#)
- [Simple Type: **trigtype**](#)
- [Simple Type: **tuning**](#)
- [Simple Type: **uuid**](#)
- [Simple Type: **velocitycurve**](#)
- [Simple Type: **vol**](#)
- [Simple Type: **waveref**](#)
- [Simple Type: **xctrl**](#)

[top](#)

Schema Document Properties

Target Namespace	None
Element and Attribute Namespaces	<ul style="list-style-type: none"> • Global element and attribute declarations belong to this schema's target namespace. • By default, local element declarations belong to this schema's target namespace. • By default, local attribute declarations belong to this schema's target namespace.
Documentation	Shared schema for all TX16Wx file types. Most objects are a fairly obvious 1<->1 mapping of the sound generators in the TX16Wx. Some attributes use "real-world" units, while some, like levels and pan(-like) values are simple 0->1 or -1->1 ranges. Almost all attributes are declared with their legal values specified, thus verifiable by the parser itself. Some legacy values exist, which are parsed relaxed and translated internally.

Declared Namespaces

Prefix	Namespace

xml	http://www.w3.org/XML/1998/namespace
xs	http://www.w3.org/2001/XMLSchema

Schema Component Representation

```
<xs:schema attributeFormDefault="qualified" elementFormDefault="qualified">
  ...
</xs:schema>
```

[top](#)

Global Definitions

Complex Type: **aeg**

Super-types: None
Sub-types: None

Name	aeg
<u>Abstract</u>	no

XML Instance Representation

```
<...
  attack="time [0..1]"
  attack-shape="pan [0..1]"
  decay1="time [0..1]"
  decay1-shape="pan [0..1]"
  level1="vol [0..1]"
  decay2="time [0..1]"
  decay2-shape="pan [0..1]"
  level2="vol [0..1]"
  sustain="vol [0..1]"
  release="time [0..1]"
  release-shape="pan [0..1]"/>
```

Schema Component Representation

```
<xs:complexType name="aeg">
  <xs:attribute name="attack" type="time"/>
  <xs:attribute name="attack-shape" type="pan"/>
  <xs:attribute name="decay1" type="time"/>
  <xs:attribute name="decay1-shape" type="pan"/>
  <xs:attribute name="level1" type="vol"/>
  <xs:attribute name="decay2" type="time"/>
  <xs:attribute name="decay2-shape" type="pan"/>
  <xs:attribute name="level2" type="vol"/>
  <xs:attribute name="sustain" type="vol"/>
  <xs:attribute name="release" type="time"/>
  <xs:attribute name="release-shape" type="pan"/>
</xs:complexType>
```

[top](#)

Complex Type: **automated-parameter**

Super-types: None

Sub-types: None

Name	automated-parameter
-------------	---------------------

Abstract	no
-----------------	----

XML Instance Representation

```
<...  
  automation="automation [0..1]"  
  midi-control="midicc [0..1]"/>
```

Schema Component Representation

```
<xs:complexType name="automated-parameter">  
  <xs:attribute name="automation" type="automation" use="optional"/>  
  <xs:attribute name="midi-control" type="midicc" use="optional"/>  
</xs:complexType>
```

[top](#)

Complex Type: automation-control

Super-types: [automation-mapping-base](#) < **automation-control** (by extension)

Sub-types: None

Name	automation-control
-------------	--------------------

Abstract	no
-----------------	----

XML Instance Representation

```
<...  
  target="uuid [1]"  
  pin="automation-pin [1]"  
  source="automation [1]"/>
```

Schema Component Representation

```
<xs:complexType name="automation-control">  
  <xs:complexContent>  
    <xs:extension base="automation-mapping-base">  
      <xs:attribute name="source" type="automation" use="required"/>  
    </xs:extension>  
  </xs:complexContent>  
</xs:complexType>
```

[top](#)

Complex Type: automation-mapping-base

Super-types: None

Sub-types:

- [midi-control](#) (by extension)
- [automation-control](#) (by extension)

Name	automation-mapping-base
<u>Abstract</u>	no

XML Instance Representation

```
<...
  target="uuid [1]"
  pin="automation-pin [1]"/>
```

Schema Component Representation

```
<xs:complexType name="automation-mapping-base">
  <xs:attribute name="target" type="uuid " use="required"/>
  <xs:attribute name="pin" type="automation-pin " use="required"/>
</xs:complexType>
```

[top](#)

Complex Type: **axis**

<i>Super-types:</i>	None
<i>Sub-types:</i>	None

Name	axis
<u>Abstract</u>	no
Documentation	Wave matrix axis. Maps a modulation source to perform wave selection.

XML Instance Representation

```
<...
  source="modsrc [1]"
  invert="xs:boolean [0..1]"
  round-robin-source="round-robin-source [0..1]">
  <curve> curve </curve> [0..1]
</...>
```

Schema Component Representation

```
<xs:complexType name="axis">
  <xs:sequence>
    <xs:element name="curve" type="curve " minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="source" type="modsrc " use="required"/>
  <xs:attribute name="invert" type="xs:boolean "/>
  <xs:attribute name="round-robin-source" type="round-robin-source "/>
</xs:complexType>
```

[top](#)

Complex Type: **bank**

<i>Super-types:</i>	versioned-element < bank (by extension)
<i>Sub-types:</i>	None

Name	bank
Abstract	no

XML Instance Representation

```

<...
created-by="pluginversion [0..1]"
compat="xs:integer [0..1]"
interpolation="interpolation [0..1] ? "
quality="quality [0..1] ? ">
  <xctrl
    ctrl="midicc [0..1]"
    offset="xs:integer [0..1]"/> [0..16]
  <midi
    program-change-mode="program-change-mode [0..1]"
    use-bank-select="xs:boolean [0..1]"/> [0..1]
  <automation
    param="automation [0..1]"
    offset="vol [0..1]"/> [0..48]
  <output
    name="xs:string [1]"
    enabled="xs:boolean [0..1]"
    stereo="xs:boolean [0..1] ? "/> [0..12]
  <wave
    path="xs:string [0..1]"/> [0..*] ?
  <wavematrix
    path="xs:string [0..1]"/> [0..*]
  <program
    path="xs:string [0..1]"/> [0..*]
  <performance
    path="xs:string [0..1]"
    active="xs:boolean [0..1]"/> [0..*]
</...>

```

Schema Component Representation

```

<xs:complexType name="bank">
  <xs:complexContent>
    <xs:extension base="versioned-element">
      <xs:sequence>
        <!-- Definition of external controllers This should move inside the "midi" element -->
        <xs:element name="xctrl" minOccurs="0" maxOccurs="16">
          <xs:complexType>
            <xs:attribute name="ctrl" type="midicc"/>
            <xs:attribute name="offset" type="xs:integer"/>
          </xs:complexType>
        </xs:element>
        <!-- MIDI settings -->
        <xs:element name="midi" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:attribute name="program-change-mode" type="program-change-mode"
              use="optional"/>
            <xs:attribute name="use-bank-select" type="xs:boolean" use="optional"/>
          </xs:complexType>
        </xs:element>
        <!-- AssignableAutomation setup -->
        <xs:element name="automation" minOccurs="0" maxOccurs="48">
          <xs:complexType>

```

```

        <xs:attribute name="param" type="automation" />
        <xs:attribute name="offset" type="vol" />
    </xs:complexType>
</xs:element>
<!-- Outputs -->
<xs:element name="output" minOccurs="0" maxOccurs="12">
    <xs:complexType>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="enabled" type="xs:boolean" use="optional"/>
        <xs:attribute name="stereo" type="xs:boolean" use="optional"/>
    </xs:complexType>
</xs:element>
<xs:element name="wave" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
        <xs:attribute name="path" type="xs:string" />
    </xs:complexType>
</xs:element>
<xs:element name="wavematrix" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
        <xs:attribute name="path" type="xs:string" />
    </xs:complexType>
</xs:element>
<!-- wave, program and performance has a minOccurs of zero (0). It is possible
through UI to save a bank with zero waves, AND zero programs. It is not possible to
save one without performances. However, if we're saving plugin state (VST
getChunk), that value can be zero as well. -->
<xs:element name="program" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
        <xs:attribute name="path" type="xs:string" />
    </xs:complexType>
</xs:element>
<xs:element name="performance" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
        <xs:attribute name="path" type="xs:string" />
        <xs:attribute name="active" type="xs:boolean" use="optional"/>
    </xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="interpolation" type="interpolation" use="optional"/>
<xs:attribute name="quality" type="quality" use="optional"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>

```

[top](#)

Complex Type: **bounds**

Super-types: None

Sub-types: None

Name	bounds
<u>Abstract</u>	no
Documentation	MIDI key/velocity range bounds

XML Instance Representation

```

<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!-- ... -->

```

```
<...
  low-key="midinote [0..1]"
  high-key="midinote [0..1]"
  low-vel="midivalue [0..1]"
  high-vel="midivalue [0..1]"
  lowkey="midinote [0..1]"
  highkey="midinote [0..1]"
  lowvel="midivalue [0..1]"
  highvel="midivalue [0..1]"/>
```

Schema Component Representation

```
<xs:complexType name="bounds">
  <xs:attribute name="low-key" type="midinote"/>
  <xs:attribute name="high-key" type="midinote"/>
  <xs:attribute name="low-vel" type="midivalue"/>
  <xs:attribute name="high-vel" type="midivalue"/>
  <xs:attribute name="lowkey" type="midinote"/>
  <xs:attribute name="highkey" type="midinote"/>
  <xs:attribute name="lowvel" type="midivalue"/>
  <xs:attribute name="highvel" type="midivalue"/>
</xs:complexType>
```

[top](#)

Complex Type: **curve**

Super-types: None
Sub-types: None

Name	curve
<u>Abstract</u>	no

XML Instance Representation

```
<...
  smooth="xs:boolean [0..1]"
  shape="curve-shape [0..1]"
  steps="sequence [0..1]"/>
```

Schema Component Representation

```
<xs:complexType name="curve">
  <xs:attribute name="smooth" type="xs:boolean"/>
  <xs:attribute name="shape" type="curve-shape"/>
  <xs:attribute name="steps" type="sequence"/>
</xs:complexType>
```

[top](#)

Complex Type: **env**

Super-types: None
Sub-types: None

Name	env
<u>Abstract</u>	no
Documentation	Modulation ENV

XML Instance Representation

```
<...
  level0="pan [0..1]"
  time1="time [0..1]"
  shape1="pan [0..1]"
  level1="pan [0..1]"
  time2="time [0..1]"
  shape2="pan [0..1]"
  level2="pan [0..1]"
  time3="time [0..1]"
  shape3="pan [0..1]"
  level3="pan [0..1]"
  amp="vol [0..1]"/>
```

Schema Component Representation

```
<xs:complexType name="env">
  <xs:attribute name="level0" type="pan"/>
  <xs:attribute name="time1" type="time"/>
  <xs:attribute name="shape1" type="pan"/>
  <xs:attribute name="level1" type="pan"/>
  <xs:attribute name="time2" type="time"/>
  <xs:attribute name="shape2" type="pan"/>
  <xs:attribute name="level2" type="pan"/>
  <xs:attribute name="time3" type="time"/>
  <xs:attribute name="shape3" type="pan"/>
  <xs:attribute name="level3" type="pan"/>
  <xs:attribute name="amp" type="vol"/>
</xs:complexType>
```

[top](#)

Complex Type: **filter**

<i>Super-types:</i>	None
<i>Sub-types:</i>	None

Name	filter
<u>Abstract</u>	no

XML Instance Representation

```
<...
  type="filtertype [1]"
  freq="freq [0..1]"
  cutoff="freq [0..1]"
  res="vol [0..1]"
  resonance="percent [0..1]"
  drive="pan [1]"
  slope="filterslope [0..1]"/>
```

Schema Component Representation

```

<xs:complexType name="filter">
  <xs:attribute name="type" type="filtertype" use="required"/>
  <xs:attribute name="freq" type="freq"/>
  <xs:attribute name="cutoff" type="freq"/>
  <xs:attribute name="res" type="vol"/>
  <xs:attribute name="resonance" type="percent"/>
  <xs:attribute name="drive" type="pan" use="required"/>
  <xs:attribute name="slope" type="filterslope" use="optional"/>
</xs:complexType>

```

[top](#)

Complex Type: **fx**

Super-types: None

Sub-types: None

Name	fx
Abstract	no

XML Instance Representation

```

<...
  type="xs:string [1]"
  bypass="xs:boolean [0..1]"
  <param> param </param> [0..*]
</...>

```

Schema Component Representation

```

<xs:complexType name="fx">
  <xs:sequence>
    <xs:element name="param" type="param" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="type" type="xs:string" use="required"/>
  <xs:attribute name="bypass" type="xs:boolean" use="optional"/>
</xs:complexType>

```

[top](#)

Complex Type: **group**

Super-types: [soundshape-base](#) < **group** (by extension)

Sub-types: None

Name	group
Abstract	no
Documentation	Group collects splits/regions (i.e. wave mappings). In TXv2 it contains sound tweaking attributes. In TXv3 it maps to a soundshape object. See reference manual for a description of the parameters.

XML Instance Representation

```

<...
  <group>
    ...
  </group>
</...>

```

```

<...
scale="scale [0..1]"
glide="time [0..1]"
velocity="pan [0..1]"
volume="vol [0..1]"
pan="pan [0..1]"
start="time [0..1]"
delay="time [0..1]"
name="xs:string [1]"
soundshape="uuid [0..1]"
color="xs:string [0..1]"
playback="playback [0..1]"
mute="mute [0..1]"
solo="solo [0..1]"
coarse="coarse [0..1]"
fine="fine [0..1]"
tuning="tuning [0..1]"
velocityOffset="midivalue [0..1]"
amp="vol [0..1]"
playmode="playmode [0..1]"
polymode="polymode [0..1]"
noteprio="noteprio [0..1]"
output="group-output [1]"
chokeGroup="chokeGroup [0..1]"
choke-group="anySimpleType [0..1]"
interpolation="interpolation [0..1]"
quality="quality [0..1]"
velocitycurve="velocitycurve [0..1]">
  <range> bounds </range> [0..1] ?
  <fade> bounds </fade> [0..1] ?
  <aeg> aeg </aeg> [0..1]
  <env1> env </env1> [0..1]
  <env2> env </env2> [0..1]
  <lfo1> lfo </lfo1> [0..1]
  <lfo2> lfo </lfo2> [0..1]
  <seq1> seq </seq1> [0..1]
  <seq2> seq </seq2> [0..1]
  <seq3> seq </seq3> [0..1]
  <filter> filter </filter> [0..1]
  <filter1> filter </filter1> [0..1]
  <filter2> filter </filter2> [0..1]
  <insert> fx </insert> [0..1]
  <send> send </send> [0..3]
  <modulation> modulation </modulation> [0..1]
  <split> split </split> [0..127]
  <region> region </region> [0..*]
  <switcher
    source="modsrc [0..1]"
    range-lo="midivalue [0..1]"
    range-hi="midivalue [0..1]"
    rand-lo="xs:float [0..1]"
    rand-hi="xs:float [0..1]"
    bpm-lo="bpm [0..1]"
    bpm-hi="bpm [0..1]"
    seq-length="xs:integer [0..1]"
    seq-pos="xs:integer [0..1]"
    key-lo="midinote [0..1]"
    key-hi="midinote [0..1]"
    key-up="midinote [0..1]"
    key-down="midinote [0..1]"

```

```

key-last="midinote [0..1]"
key-prev="midinote [0..1]"
round-robin-source="round-robin-source [0..1]"
enabled="xs:boolean [0..1]"/> [0..1] ?

```

```
</...>
```

Schema Component Representation

```

<xs:complexType name="group">
  <xs:complexContent>
    <xs:extension base="soundshape-base">
      <xs:sequence>
        <!-- key / vel range -->
        <xs:element name="range" type="bounds" maxOccurs="1" minOccurs="0"/>
        <xs:element name="fade" type="bounds" maxOccurs="1" minOccurs="0"/>
        <xs:group ref="sound-elements"/>
        <xs:element name="split" type="split" minOccurs="0" maxOccurs="127"/>
        <xs:element name="region" type="region" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:element name="switcher" maxOccurs="1" minOccurs="0">
        <xs:complexType>
          <xs:attribute name="source" type="modsrc"/>
          <xs:attribute name="range-lo" type="midivalue"/>
          <xs:attribute name="range-hi" type="midivalue"/>
          <xs:attribute name="rand-lo" type="xs:float"/>
          <xs:attribute name="rand-hi" type="xs:float"/>
          <xs:attribute name="bpm-lo" type="bpm"/>
          <xs:attribute name="bpm-hi" type="bpm"/>
          <xs:attribute name="seq-length" type="xs:integer"/>
          <xs:attribute name="seq-pos" type="xs:integer"/>
          <xs:attribute name="key-lo" type="midinote"/>
          <xs:attribute name="key-hi" type="midinote"/>
          <xs:attribute name="key-up" type="midinote"/>
          <xs:attribute name="key-down" type="midinote"/>
          <xs:attribute name="key-last" type="midinote"/>
          <xs:attribute name="key-prev" type="midinote"/>
          <xs:attribute name="round-robin-source" type="round-robin-source"/>
          <xs:attribute name="enabled" type="xs:boolean"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <!-- See manual for these attributes -->
    <xs:attribute name="name" type="xs:string" use="required"/>
    <!-- TXv3 -->
    <xs:attribute name="soundshape" type="uuid"/>
    <!-- TXv3 -->
    <xs:attribute name="color" type="xs:string"/>
    <!-- TXv3 -->
    <xs:attribute name="playback" type="playback"/>
    <xs:attribute name="mute" type="mute"/>
    <xs:attribute name="solo" type="solo"/>
    <xs:attribute name="coarse" type="coarse"/>
    <xs:attribute name="fine" type="fine"/>
    <xs:attribute name="tuning" type="tuning"/>
    <!-- TXv2 only -->
    <xs:attribute name="velocityOffset" type="midivalue"/>
    <xs:attribute name="amp" type="vol"/>
    <xs:attribute name="playmode" type="playmode"/>
    <xs:attribute name="polymode" type="polymode"/>
    <xs:attribute name="noteprio" type="noteprio"/>
  </xs:extension>
</xs:complexType>

```

```

<xs:attribute name="output" type="group-output" use="required"/>
<xs:attribute name="chokeGroup" type="chokeGroup"/>
<xs:attribute name="choke-group"/>
<xs:attribute name="interpolation" type="interpolation"/>
<xs:attribute name="quality" type="quality"/>
<!-- TXv2 only -->
<xs:attribute name="velocitycurve" type="velocitycurve"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>

```

[top](#)

Complex Type: **id-ref**

Super-types: None

Sub-types:

- [wave](#) (by extension)

Name	id-ref
Abstract	no
Documentation	Base type for wave/matrix references

XML Instance Representation

```

<...
  id="xs:integer [1]"
  path="xs:string [1]"/>

```

Schema Component Representation

```

<xs:complexType name="id-ref">
  <xs:attribute name="id" type="xs:integer" use="required"/>
  <xs:attribute name="path" type="xs:string" use="required"/>
</xs:complexType>

```

[top](#)

Complex Type: **lfo**

Super-types: [lfobase](#) < **lfo** (by extension)

Sub-types: None

Name	lfo
Abstract	no

XML Instance Representation

```

<...
  rate="freq [1]"
  sync="synctype [1]"
  fadein="time [0..1]"
  fade-in="time [0..1]"
  pos="vol [0..1]"

```

```

amp="vol [0..1]"
mode="lfomode [0..1]"
trig="trigtype [0..1]"
type="lftype [1]">

```

Schema Component Representation

```

<xs:complexType name="lfo">
  <xs:complexContent>
    <xs:extension base="lfobase">
      <xs:attribute name="type" type="lftype" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

[top](#)

Complex Type: lfobase

Super-types: None

Sub-types:

- [lfo](#) (by extension)
- [seq](#) (by extension)

Name	lfobase
<u>Abstract</u>	no
Documentation	Super type of LFO & SEQ

XML Instance Representation

```

<...
rate="freq [1]"
sync="synctype [1]"
fadein="time [0..1]"
fade-in="time [0..1]"
pos="vol [0..1]"
amp="vol [0..1]"
mode="lfomode [0..1]"
trig="trigtype [0..1]">

```

Schema Component Representation

```

<xs:complexType name="lfobase">
  <xs:attribute name="rate" type="freq" use="required"/>
  <xs:attribute name="sync" type="synctype" use="required"/>
  <xs:attribute name="fadein" type="time" use="optional"/>
  <xs:attribute name="fade-in" type="time" use="optional"/>
  <xs:attribute name="pos" type="vol" use="optional"/>
  <xs:attribute name="amp" type="vol" use="optional"/>
  <xs:attribute name="mode" type="lfomode" use="optional"/>
  <!-- Optional legacy -->
  <xs:attribute name="trig" type="trigtype" use="optional"/>
</xs:complexType>

```

[top](#)

Complex Type: loop

Super-types: None

Sub-types: None

Name	loop
<u>Abstract</u>	no
Documentation	Wave loop

XML Instance Representation

```
<...  
  name="xs:string [0..1]"  
  start="xs:integer [1]"  
  end="xs:integer [1]"  
  mode="loopmode [1]"/>
```

Schema Component Representation

```
<xs:complexType name="loop">  
  <xs:attribute name="name" type="xs:string" use="optional"/>  
  <xs:attribute name="start" type="xs:integer" use="required"/>  
  <xs:attribute name="end" type="xs:integer" use="required"/>  
  <xs:attribute name="mode" type="loopmode" use="required"/>  
</xs:complexType>
```

[top](#)

Complex Type: **mapping**

Super-types: None

Sub-types:

- [region](#) (by extension)
- [split](#) (by extension)

Name	mapping
<u>Abstract</u>	no

XML Instance Representation

```
<...  
  wave="waveref [0..1]"  
  matrix="waveref [0..1]"  
  loop="xs:integer [0..1]"  
  release="xs:integer [0..1]"  
  root="rootkey [0..1]"  
  fine="fine [0..1]"  
  mode="xs:string [0..1]"  
  attenuation="vol [0..1]"  
  pan="pan [0..1]"  
  shift="coarse [0..1]"  
  tune="fine [0..1]"/>
```

Schema Component Representation

```
<xs:complexType name="mapping">  
  <xs:attribute name="wave" type="waveref"/>
```

```

<xs:attribute name="matrix" type="waveref" />
<xs:attribute name="loop" type="xs:integer" />
<xs:attribute name="release" type="xs:integer" />
<xs:attribute name="root" type="rootkey" />
<xs:attribute name="fine" type="fine" />
<xs:attribute name="mode" type="xs:string" />
<xs:attribute name="attenuation" type="vol" />
<xs:attribute name="pan" type="pan" />
<xs:attribute name="shift" type="coarse" />
<xs:attribute name="tune" type="fine" />
</xs:complexType>

```

[top](#)

Complex Type: **midi-control**

Super-types: [automation-mapping-base](#) < **midi-control** (by extension)

Sub-types: None

Name	midi-control
<u>Abstract</u>	no

XML Instance Representation

```

<...
  target="uuid [1]"
  pin="automation-pin [1]"
  source="midicc [1]"
  program="xs:string [0..1]"/>

```

Schema Component Representation

```

<xs:complexType name="midi-control">
  <xs:complexContent>
    <xs:extension base="automation-mapping-base">
      <xs:attribute name="source" type="midicc" use="required"/>
      <xs:attribute name="program" type="xs:string" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

[top](#)

Complex Type: **modulation**

Super-types: None

Sub-types: None

Name	modulation
<u>Abstract</u>	no
Documentation	Modulation matrix

XML Instance Representation

```

<...
  modulation="..."
  ...
>

```



```

<...>
  <entry
    source="modsrc [1]"
    destination="moddst [1]"
    via="modsrc [0..1]"
    amount="xs:string [0..1]"
    amount_via="xs:string [0..1]"
    frozen="xs:boolean [0..1]"
    enabled="xs:boolean [0..1]"> [0..16]
    <src-curve> curve </src-curve> [0..1]
    <via-curve> curve </via-curve> [0..1]
  </entry>
</...>

```

Schema Component Representation

```

<xs:complexType name="modulation">
  <xs:sequence>
    <xs:element name="entry" maxOccurs="16" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="src-curve" type="curve" maxOccurs="1" minOccurs="0"/>
          <xs:element name="via-curve" type="curve" maxOccurs="1" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="source" type="modsrc" use="required"/>
        <xs:attribute name="destination" type="moddst" use="required"/>
        <xs:attribute name="via" type="modsrc" />
        <xs:attribute name="amount" type="xs:string" />
        <xs:attribute name="amount_via" type="xs:string" />
        <xs:attribute name="frozen" type="xs:boolean" />
        <xs:attribute name="enabled" type="xs:boolean" />
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

```

[top](#)

Complex Type: **param**

Super-types: None

Sub-types: None

Name	param
<u>Abstract</u>	no

XML Instance Representation

```

<...
  name="xs:string [1]"
  value="xs:string [1]"
  automation="automation [0..1]"/>

```

Schema Component Representation

```

<xs:complexType name="param">
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="value" type="xs:string" use="required"/>

```

```
<xs:attribute name="automation" type="automation" use="optional"/>
</xs:complexType>
```

[top](#)

Complex Type: **performance**

Super-types: [versioned-element](#) < **performance** (by extension)

Sub-types: None

Name	performance
<u>Abstract</u>	no
Documentation	Maps slots and associated programs.

XML Instance Representation

```
<...
created-by="pluginversion [0..1]"
compat="xs:integer [0..1]"
name="xs:string [0..1]">
  <slot> slot </slot> [0..*]
  <fx
    type="xs:string [1]"
    bypass="xs:boolean [0..1]"
    bus="fx-bus [1]"
    level="vol [0..1]"
    wet="percent [0..1]"
    output="send-destination [1]"
    mute="xs:boolean [0..1]"> [0..6]
    <param> param </param> [0..*]
  </fx>
  <midi-control> midi-control </midi-control> [0..*] ?
  <automation> automation-control </automation> [0..*] ?
</...>
```

Schema Component Representation

```
<xs:complexType name="performance">
  <xs:complexContent>
    <xs:extension base="versioned-element">
      <xs:sequence>
        <xs:element name="slot" type="slot" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="fx" minOccurs="0" maxOccurs="6">
          <xs:complexType>
            <xs:complexContent>
              <xs:extension base="fx">
                <xs:attribute name="bus" type="fx-bus" use="required"/>
                <xs:attribute name="level" type="vol" use="optional"/>
                <xs:attribute name="wet" type="percent" use="optional"/>
                <xs:attribute name="output" type="send-destination" use="required"/>
              </xs:extension>
            </xs:complexContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

<-- TXv3 automation -->

```

<xs:element name="midi-control" type="midi-control" minOccurs="0" maxOccurs="
unbounded"/>
<xs:element name="automation" type="automation-control" minOccurs="0"
maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="name" type="xs:string" />
</xs:extension>
</xs:complexContent>
</xs:complexType>

```

[top](#)

Complex Type: **program**

Super-types: [versioned-element](#) < **program** (by extension)

Sub-types: None

Name	program
<u>Abstract</u>	no

XML Instance Representation

```

<...
created-by="pluginversion [0..1]"
compat="xs:integer [0..1]"
name="xs:string [1]"
interpolation="interpolation [0..1]"
icon="xs:string [0..1]"
quality="quality [0..1]">
  Start Choice [0..*] ?
    <wave> wave </wave> [0..*]
    <wavematrix> id-ref </wavematrix> [0..*]
  End Choice
  Start Choice [1]
    <range> bounds </range> [1]
    <bounds> bounds </bounds> [1]
  End Choice
  <soundshape> soundshape </soundshape> [0..*]
  <group> group </group> [0..*]
</...>

```

Schema Component Representation

```

<xs:complexType name="program">
  <xs:complexContent>
    <xs:extension base="versioned-element">
      <xs:sequence>
        <xs:choice minOccurs="0" maxOccurs="unbounded">
          <xs:element name="wave" type="wave" minOccurs="0" maxOccurs="unbounded"
"/>
          <xs:element name="wavematrix" type="id-ref" minOccurs="0" maxOccurs="
unbounded"/>
        </xs:choice>
        <xs:choice maxOccurs="1">
          <!-- TXv2 -->
          <xs:element name="range" type="bounds" />
          <xs:element name="bounds" type="bounds" />
        </xs:choice>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

```

    <xs:element name="soundshape" type="soundshape" minOccurs="0" maxOccurs="
unbounded"/>
    <xs:element name="group" type="group" minOccurs="0" maxOccurs="unbounded"
/>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="interpolation" type="interpolation" use="optional"/>
  <xs:attribute name="icon" type="xs:string"/>
  <xs:attribute name="quality" type="quality"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>

```

[top](#)

Complex Type: **region**

Super-types: [mapping](#) < **region** (by extension)

Sub-types: None

Name	region
Abstract	no

XML Instance Representation

```

<...
  wave="waveref [0..1]"
  matrix="waveref [0..1]"
  loop="xs:integer [0..1]"
  release="xs:integer [0..1]"
  root="rootkey [0..1]"
  fine="fine [0..1]"
  mode="xs:string [0..1]"
  attenuation="vol [0..1]"
  pan="pan [0..1]"
  shift="coarse [0..1]"
  tune="fine [0..1]"
  mute="xs:boolean [0..1]"
  <bounds> bounds </bounds> [1]
  <fade> bounds </fade> [0..1]
</...>

```

Schema Component Representation

```

<xs:complexType name="region">
  <xs:complexContent>
    <xs:extension base="mapping">
      <xs:sequence>
        <xs:element name="bounds" type="bounds"/>
        <xs:element name="fade" type="bounds" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="mute" type="xs:boolean"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

[top](#)

Complex Type: **send**

Super-types: None

Sub-types: None

Name	send
<u>Abstract</u>	no

XML Instance Representation

```
<...  
  dest="send-destination [0..1]"  
  level="vol [0..1]"  
  pre-fader="xs:boolean [0..1]"/>
```

Schema Component Representation

```
<xs:complexType name="send">  
  <xs:attribute name="dest" type="send-destination " use="optional"/>  
  <xs:attribute name="level" type="vol " use="optional"/>  
  <xs:attribute name="pre-fader" type="xs:boolean " use="optional"/>  
</xs:complexType>
```

[top](#)

Complex Type: **seq**

Super-types: [lfobase](#) < **seq** (by extension)

Sub-types: None

Name	seq
<u>Abstract</u>	no
Documentation	Step sequence LFO modulator. LFO using user defined patterns.

XML Instance Representation

```
<...  
  rate="freq [1]"  
  sync="synctype [1]"  
  fadein="time [0..1]"  
  fade-in="time [0..1]"  
  pos="vol [0..1]"  
  amp="vol [0..1]"  
  mode="lfomode [0..1]"  
  trig="trigtype [0..1]"  
  smooth="xs:boolean [0..1]"  
  stepsync="xs:boolean [0..1]"  
  values="sequence [0..1]"  
  length="seqlength [1]"/>
```

Schema Component Representation

```
<xs:complexType name="seq">  
  <xs:complexContent>  
    <xs:extension base="lfobase ">
```

```

<xs:attribute name="smooth" type="xs:boolean"/>
<xs:attribute name="stepsync" type="xs:boolean"/>
<xs:attribute name="values" type="sequence"/>
<xs:attribute name="length" type="seqlength" use="required"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>

```

[top](#)

Complex Type: **slot**

Super-types: None

Sub-types: None

Name	slot
Abstract	no

XML Instance Representation

```

<...
name="xs:string [0..1]"
channel="channel [1]"
program="xs:string [1]"
output="output [0..1]"
pan="pan [0..1]"
amp="vol [0..1]"
volume="vol [0..1]"
transpose="coarse [0..1]"
detune="fine [0..1]"
poly="poly [0..1]"
solo="solo [0..1]"
mute="mute [0..1]"
interpolation="interpolation [0..1]"
velocitycurve="velocitycurve [0..1]"
color="xs:string [0..1]">
  <velocity-curve> curve </velocity-curve> [0..1]
  <amp
    automation="automation [0..1]"
    midi-control="midicc [0..1]"
    value="vol [1]"/> [0..1]
  <volume
    automation="automation [0..1]"
    midi-control="midicc [0..1]"
    value="vol [1]"/> [0..1]
  <pan
    automation="automation [0..1]"
    midi-control="midicc [0..1]"
    value="pan [1]"/> [0..1]
  <transpose
    automation="automation [0..1]"
    midi-control="midicc [0..1]"
    value="coarse [1]"/> [0..1]
  <detune
    automation="automation [0..1]"
    midi-control="midicc [0..1]"
    value="fine [1]"/> [0..1]

```

```

<send
  dest="send-destination [0..1]"
  level="vol [0..1]"
  pre-fader="xs:boolean [0..1]"> [0..3] ?
    <level
      automation="automation [0..1]"
      midi-control="midicc [0..1]"
      value="vol [1]"/> [0..1]
    </send>
  <arpeggiator
    enabled="xs:boolean [0..1]"
    pattern="xs:integer [0..1]"> [0..1] ?
      <pattern
        number="xs:integer [0..1]"
        size="xs:integer [0..1]"
        rate="xs:string [0..1]"
        mode="xs:string [0..1]"
        octave="xs:integer [0..1]"
        length="percent [0..1]"
        strum="percent [0..1]"
        shuffle="percent [0..1]"> [0..8]
          <step
            on="xs:boolean [0..1]"
            length="xs:integer [0..1]"
            octave="xs:integer [0..1]"
            velocity="midivalue [0..1]"
            semi="xs:integer [0..1]"/> [0..32]
          </step>
        </pattern>
      </arpeggiator>
    <midi-control> midi-control </midi-control> [0..*] ?
    <automation> automation-control </automation> [0..*] ?
  </...>

```

Schema Component Representation

```

<xs:complexType name="slot">
  <xs:sequence>
    <xs:element name="velocity-curve" type="curve" minOccurs="0" maxOccurs="1"/>
    <!-- TXv2 legacy name -->
    <xs:element name="amp" maxOccurs="1" minOccurs="0">
      <xs:complexType>
        <xs:complexContent>
          <xs:extension base="automated-parameter">
            <xs:attribute name="value" type="vol" use="required"/>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="volume" maxOccurs="1" minOccurs="0">
      <xs:complexType>
        <xs:complexContent>
          <xs:extension base="automated-parameter">
            <xs:attribute name="value" type="vol" use="required"/>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="pan" maxOccurs="1" minOccurs="0">
      <xs:complexType>
        <xs:complexContent>
          <xs:extension base="automated-parameter">

```

```

        <xs:attribute name="value" type="pan" use="required"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="transpose" maxOccurs="1" minOccurs="0">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="automated-parameter">
        <xs:attribute name="value" type="coarse" use="required"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="detune" maxOccurs="1" minOccurs="0">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="automated-parameter">
        <xs:attribute name="value" type="fine" use="required"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="send" maxOccurs="3" minOccurs="0">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="send">
        <xs:sequence>
          <xs:element name="level" maxOccurs="1" minOccurs="0">
            <xs:complexType>
              <xs:complexContent>
                <xs:extension base="automated-parameter">
                  <xs:attribute name="value" type="vol" use="required"/>
                </xs:extension>
              </xs:complexContent>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="arpeggiator" maxOccurs="1" minOccurs="0">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="pattern" minOccurs="0" maxOccurs="8">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="step" minOccurs="0" maxOccurs="32">
              <xs:complexType>
                <xs:attribute name="on" type="xs:boolean" use="optional"/>
                <xs:attribute name="length" type="xs:integer" use="optional"/>
                <xs:attribute name="octave" type="xs:integer" use="optional"/>
                <xs:attribute name="velocity" type="midivalue" use="optional"/>
                <xs:attribute name="semi" type="xs:integer" use="optional"/>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
          <xs:attribute name="number" type="xs:integer" use="optional"/>
          <xs:attribute name="size" type="xs:integer" use="optional"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```



```

        <xs:attribute name="rate" type="xs:string" use="optional"/>
        <xs:attribute name="mode" type="xs:string" use="optional"/>
        <xs:attribute name="octave" type="xs:integer" use="optional"/>
        <xs:attribute name="length" type="percent" use="optional"/>
        <xs:attribute name="strum" type="percent" use="optional"/>
        <xs:attribute name="shuffle" type="percent" use="optional"/>
    </xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="enabled" type="xs:boolean" use="optional"/>
<xs:attribute name="pattern" type="xs:integer" use="optional"/>
</xs:complexType>
</xs:element>
<xs:element name="midi-control" type="midi-control" minOccurs="0" maxOccurs="
unbounded"/>
<xs:element name="automation" type="automation-control" minOccurs="0" maxOccurs="
unbounded"/>
</xs:sequence>
<xs:attribute name="name" type="xs:string" use="optional"/>
<xs:attribute name="channel" type="channel" use="required"/>
<xs:attribute name="program" type="xs:string" use="required"/>
<xs:attribute name="output" type="output"/>
<xs:attribute name="pan" type="pan" use="optional"/>
<xs:attribute name="amp" type="vol" use="optional"/>
<xs:attribute name="volume" type="vol" use="optional"/>
<xs:attribute name="transpose" type="coarse" use="optional"/>
<xs:attribute name="detune" type="fine" use="optional"/>
<xs:attribute name="poly" type="poly" use="optional"/>
<xs:attribute name="solo" type="solo" use="optional"/>
<xs:attribute name="mute" type="mute" use="optional"/>
<xs:attribute name="interpolation" type="interpolation" use="optional"/>
<xs:attribute name="velocitycurve" type="velocitycurve" use="optional"/>
<xs:attribute name="color" type="xs:string"/>
</xs:complexType>

```

[top](#)

Complex Type: **soundshape**

Super-types: [soundshape-base](#) < **soundshape** (by extension)

Sub-types: None

Name	soundshape
<u>Abstract</u>	no

XML Instance Representation

```

<...
  name="xs:string [0..1]"
  coarse="coarse [0..1]"
  fine="fine [0..1]"
  tuning="tuning [0..1]"
  scale="scale [0..1]"
  glide="time [0..1]"
  velocity="pan [0..1]"
  volume="vol [0..1]"
  pan="pan [0..1]"
  start="time [0..1]"

```

```

delay="time [0..1]"
glide-mode="glide-mode [0..1]"
id="uuid [1]"
velocity-offset="midivalue [1]"
velocity-curve="xs:string [0..1]">
  <aeg> aeg </aeg> [0..1]
  <env1> env </env1> [0..1]
  <env2> env </env2> [0..1]
  <lfo1> lfo </lfo1> [0..1]
  <lfo2> lfo </lfo2> [0..1]
  <seq1> seq </seq1> [0..1]
  <seq2> seq </seq2> [0..1]
  <seq3> seq </seq3> [0..1]
  <filter> filter </filter> [0..1]
  <filter1> filter </filter1> [0..1]
  <filter2> filter </filter2> [0..1]
  <insert> fx </insert> [0..1]
  <send> send </send> [0..3]
  <modulation> modulation </modulation> [0..1]
  <velocity-curve> curve </velocity-curve> [0..1]
</...>

```

Schema Component Representation

```

<xs:complexType name="soundshape">
  <xs:complexContent>
    <xs:extension base="soundshape-base">
      <xs:sequence>
        <xs:group ref="sound-elements"/>
        <xs:element name="velocity-curve" type="curve" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:attribute name="glide-mode" type="glide-mode"/>
      <xs:attribute name="id" type="uuid" use="required"/>
      <xs:attribute name="velocity-offset" type="midivalue" use="required"/>
      <xs:attribute name="velocity-curve" type="xs:string"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

[top](#)

Complex Type: soundshape-base

Super-types: None

Sub-types:

- [soundshape](#) (by extension)
- [group](#) (by extension)

Name	soundshape-base
Abstract	no

XML Instance Representation

```

<...
name="xs:string [0..1]"
coarse="coarse [0..1]"
fine="fine [0..1]"
tuning="tuning [0..1]"

```

```

scale="scale [0..1]"
glide="time [0..1]"
velocity="pan [0..1]"
volume="vol [0..1]"
pan="pan [0..1]"
start="time [0..1]"
delay="time [0..1]"/>

```

Schema Component Representation

```

<xs:complexType name="soundshape-base">
  <!-- See manual for these attributes -->
  <xs:attribute name="name" type="xs:string"/>
  <xs:attribute name="coarse" type="coarse"/>
  <xs:attribute name="fine" type="fine"/>
  <xs:attribute name="tuning" type="tuning"/>
  <xs:attribute name="scale" type="scale"/>
  <xs:attribute name="glide" type="time"/>
  <xs:attribute name="velocity" type="pan"/>
  <xs:attribute name="volume" type="vol"/>
  <xs:attribute name="pan" type="pan"/>
  <xs:attribute name="start" type="time"/>
  <xs:attribute name="delay" type="time"/>
</xs:complexType>

```

[top](#)

Complex Type: **split**

Super-types: [mapping](#) < **split** (by extension)

Sub-types: None

Name	split
<u>Abstract</u>	no
Documentation	A split maps a wave or a wavematrix to a key range within a group. It also assigns independant sustain and release loop, and allows for root/tune override.

XML Instance Representation

```

<...
wave="waveref [0..1]"
matrix="waveref [0..1]"
loop="xs:integer [0..1]"
release="xs:integer [0..1]"
root="rootkey [0..1]"
fine="fine [0..1]"
mode="xs:string [0..1]"
attenuation="vol [0..1]"
pan="pan [0..1]"
shift="coarse [0..1]"
tune="fine [0..1]"
key="midinote [1]"/>

```

Schema Component Representation

```

<xs:complexType name="split">
  <xs:complexContent>

```

```

<xs:extension base="mapping">
  <xs:attribute name="key" type="midinote" use="required"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>

```

[top](#)

Complex Type: **versioned-element**

Super-types: None

Sub-types:

- [wavematrix](#) (by extension)
- [program](#) (by extension)
- [performance](#) (by extension)
- [bank](#) (by extension)

Name	versioned-element
Abstract	no

XML Instance Representation

```

<...
  created-by="pluginversion [0..1]"
  compat="xs:integer [0..1]"/>

```

Schema Component Representation

```

<xs:complexType name="versioned-element">
  <!-- which version of TX16Wx created this - used for legacy detection -->
  <xs:attribute name="created-by" type="pluginversion" use="optional"/>
  <xs:attribute name="compat" type="xs:integer" use="optional"/>
</xs:complexType>

```

[top](#)

Complex Type: **wave**

Super-types: [id-ref](#) < **wave** (by extension)

Sub-types: None

Name	wave
Abstract	no
Documentation	Wave reference. Path can be file- or "places"-relative or absolute. Also allows overriding some meta data of the wave.

XML Instance Representation

```

<...
  id="xs:integer [1]"
  path="xs:string [1]"
  start="xs:integer [0..1]"
  end="xs:integer [0..1]"
  root="rootkey [0..1]"

```

```

fine="fine [0..1]"
tempo="tempo [0..1]">
  <loop> loop </loop> [0..*]
</...>

```

Schema Component Representation

```

<xs:complexType name="wave">
  <xs:complexContent>
    <xs:extension base="id-ref">
      <xs:sequence>
        <xs:element name="loop" type="loop" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="start" type="xs:integer" use="optional"/>
      <xs:attribute name="end" type="xs:integer" use="optional"/>
      <xs:attribute name="root" type="rootkey" use="optional"/>
      <xs:attribute name="fine" type="fine" use="optional"/>
      <xs:attribute name="tempo" type="tempo" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

[top](#)

Complex Type: **wavematrix**

Super-types: [versioned-element](#) < **wavematrix** (by extension)

Sub-types: None

Name	wavematrix
<u>Abstract</u>	no

XML Instance Representation

```

<...
name="xs:string [1]"
created-by="pluginversion [0..1]"
compat="xs:integer [0..1]">
  <wave> wave </wave> [0..*] ?
  <xaxis> axis </xaxis> [1]
  <yaxis> axis </yaxis> [1]
  <column
    value="midivalue [1]"/> [0..126] ?
  <row
    value="midivalue [1]"> [1..127] ?
    <cell> mapping </cell> [1..127]
  </row>
</...>

```

Schema Component Representation

```

<xs:complexType name="wavematrix">
  <xs:complexContent>
    <xs:extension base="versioned-element">
      <xs:sequence>
        <xs:element name="wave" type="wave" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="xaxis" type="axis" minOccurs="1" maxOccurs="1"/>

```

```

<xs:element name="yaxis" type="axis" minOccurs="1" maxOccurs="1"/>
<xs:element name="column" minOccurs="0" maxOccurs="126">
  <xs:complexType>
    <xs:attribute name="value" type="midivalue" use="required"/>
  </xs:complexType>
</xs:element>
<xs:element name="row" minOccurs="1" maxOccurs="127">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="cell" type="mapping" minOccurs="1" maxOccurs="127"/>
    </xs:sequence>
    <xs:attribute name="value" type="midivalue" use="required"/>
  </xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="name" type="xs:string" use="required"/>
<!-- which version of TX16Wx created this - used for legacy detection -->
<xs:attribute name="created-by" type="pluginversion" use="optional"/>
<xs:attribute name="compat" type="xs:integer" use="optional"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>

```

[top](#)

Model Group: sound-elements

Name	sound-elements
------	----------------

XML Instance Representation

```

<aeg> aeg </aeg> [0..1]
<env1> env </env1> [0..1]
<env2> env </env2> [0..1]
<lfo1> lfo </lfo1> [0..1]
<lfo2> lfo </lfo2> [0..1]
<seq1> seq </seq1> [0..1]
<seq2> seq </seq2> [0..1]
<seq3> seq </seq3> [0..1]
<filter> filter </filter> [0..1]
<filter1> filter </filter1> [0..1]
<filter2> filter </filter2> [0..1]
<insert> fx </insert> [0..1]
<send> send </send> [0..3]
<modulation> modulation </modulation> [0..1]

```

Schema Component Representation

```

<xs:group name="sound-elements">
  <xs:sequence>
    <xs:element name="aeg" type="aeg" minOccurs="0" maxOccurs="1"/>
    <xs:element name="env1" type="env" minOccurs="0" maxOccurs="1"/>
    <xs:element name="env2" type="env" minOccurs="0" maxOccurs="1"/>
    <xs:element name="lfo1" type="lfo" minOccurs="0" maxOccurs="1"/>
    <xs:element name="lfo2" type="lfo" minOccurs="0" maxOccurs="1"/>
    <!-- optional (for old banks) -->
    <xs:element name="seq1" type="seq" maxOccurs="1" minOccurs="0"/>
    <xs:element name="seq2" type="seq" maxOccurs="1" minOccurs="0"/>
    <xs:element name="seq3" type="seq" maxOccurs="1" minOccurs="0"/>
  </xs:sequence>
</xs:group>

```

```

<xs:element name="filter" type="filter" minOccurs="0" maxOccurs="1"/>
<xs:element name="filter1" type="filter" minOccurs="0" maxOccurs="1"/>
<xs:element name="filter2" type="filter" minOccurs="0" maxOccurs="1"/>
<-- modulators -->
<xs:element name="insert" type="fx" maxOccurs="1" minOccurs="0"/>
<xs:element name="send" type="send" maxOccurs="3" minOccurs="0"/>
<xs:element name="modulation" type="modulation" maxOccurs="1" minOccurs="0"/>
</xs:sequence>
</xs:group>

```

[top](#)

Simple Type: **automation**

Super-types: [xs:string](#) < **automation** (by restriction)

Sub-types: None

Name	automation
Content	<ul style="list-style-type: none"> Base XSD Type: string <i>pattern</i> = Param ([0-9][0-9] 1[0-1][0-9] 12[0-7]) [0-9]+

Schema Component Representation

```

<xs:simpleType name="automation">
  <xs:restriction base="xs:string">
    <xs:pattern value="Param ([0-9][0-9]|1[0-1][0-9]|12[0-7])|[0-9]+"/>
  </xs:restriction>
</xs:simpleType>

```

[top](#)

Simple Type: **automation-pin**

Super-types: [xs:string](#) < **automation-pin** (by restriction)

Sub-types: None

Name	automation-pin
Content	<ul style="list-style-type: none"> Base XSD Type: string

Schema Component Representation

```

<xs:simpleType name="automation-pin">
  <xs:restriction base="xs:string">
    <-- TODO -->
  </xs:restriction>
</xs:simpleType>

```

[top](#)

Simple Type: **bpm**

Super-types: [xs:integer](#) < **bpm** (by restriction)

Sub-types: None

Name	bpm
Content	<ul style="list-style-type: none">Base XSD Type: integer$0 \leq \text{value} \leq 500$

Schema Component Representation

```
<xs:simpleType name="bpm">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="0"/>
    <xs:maxInclusive value="500"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **channel**

Super-types: [xs:integer](#) < **channel** (by restriction)

Sub-types: None

Name	channel
Content	<ul style="list-style-type: none">Base XSD Type: integer$0 \leq \text{value} \leq 16$

Schema Component Representation

```
<xs:simpleType name="channel">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="0"/>
    <xs:maxInclusive value="16"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **chokeGroup**

Super-types: [xs:integer](#) < **chokeGroup** (by restriction)

Sub-types: None

Name	chokeGroup
Content	<ul style="list-style-type: none">Base XSD Type: integer$0 \leq \text{value} \leq 8$

Schema Component Representation

```
<xs:simpleType name="chokeGroup">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="0"/>
    <xs:maxInclusive value="8"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **coarse**

Super-types: [xs:integer](#) < **coarse** (by restriction)

Sub-types: None

Name	coarse
Content	<ul style="list-style-type: none">Base XSD Type: integer-127 <= <i>value</i> <= 127

Schema Component Representation

```
<xs:simpleType name="coarse">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="-127"/>
    <xs:maxInclusive value="127"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **curve-shape**

Super-types: [xs:string](#) < **curve-shape** (by restriction)

Sub-types: None

Name	curve-shape
Content	<ul style="list-style-type: none">Base XSD Type: string<i>value</i> comes from list: {'Default' 'Linear' 'Exponential 1' 'Exponential 2' 'Exponential 3' 'Logarithmic 1' 'Logarithmic 2' 'Logarithmic 3' 'Inverse' 'Inverse Exponential 1' 'Inverse Exponential 2' 'Inverse Exponential 3' 'Inverse Logarithmic 1' 'Inverse Logarithmic 2' 'Inverse Logarithmic 3' 'User'}
Documentation	TXv3 only

Schema Component Representation

```
<xs:simpleType name="curve-shape">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Default"/>
  </xs:restriction>
</xs:simpleType>
```

```

<xs:enumeration value="Linear"/>
<xs:enumeration value="Exponential 1"/>
<xs:enumeration value="Exponential 2"/>
<xs:enumeration value="Exponential 3"/>
<xs:enumeration value="Logarithmic 1"/>
<xs:enumeration value="Logarithmic 2"/>
<xs:enumeration value="Logarithmic 3"/>
<xs:enumeration value="Inverse"/>
<xs:enumeration value="Inverse Exponential 1"/>
<xs:enumeration value="Inverse Exponential 2"/>
<xs:enumeration value="Inverse Exponential 3"/>
<xs:enumeration value="Inverse Logarithmic 1"/>
<xs:enumeration value="Inverse Logarithmic 2"/>
<xs:enumeration value="Inverse Logarithmic 3"/>
<xs:enumeration value="User"/>
</xs:restriction>
</xs:simpleType>

```

[top](#)

Simple Type: **filterslope**

Super-types: [xs:string](#) < **filterslope** (by restriction)

Sub-types: None

Name	filterslope
Content	<ul style="list-style-type: none"> Base XSD Type: string <i>value</i> comes from list: {'24dB' '12dB' '6dB'}

Schema Component Representation

```

<xs:simpleType name="filterslope">
  <xs:restriction base="xs:string">
    <xs:enumeration value="24dB"/>
    <xs:enumeration value="12dB"/>
    <xs:enumeration value="6dB"/>
  </xs:restriction>
</xs:simpleType>

```

[top](#)

Simple Type: **filtertype**

Super-types: [xs:string](#) < **filtertype** (by restriction)

Sub-types: None

Name	filtertype
Content	<ul style="list-style-type: none"> Base XSD Type: string <i>value</i> comes from list: {'Off' 'AllPass' 'LowPass' 'HighPass' 'BandPass' 'Notch' 'LowShelf' 'HighShelf' 'Peak'}

Schema Component Representation

```
<xs:simpleType name="filtertype">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Off"/>
    <xs:enumeration value="AllPass"/>
    <xs:enumeration value="LowPass"/>
    <xs:enumeration value="HighPass"/>
    <xs:enumeration value="BandPass"/>
    <xs:enumeration value="Notch"/>
    <xs:enumeration value="LowShelf"/>
    <xs:enumeration value="HighShelf"/>
    <xs:enumeration value="Peak"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **fine**

Super-types: [xs:integer](#) < **fine** (by restriction)

Sub-types: None

Name	fine
Content	<ul style="list-style-type: none">Base XSD Type: integer-50 <= <i>value</i> <= 50

Schema Component Representation

```
<xs:simpleType name="fine">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="-50"/>
    <xs:maxInclusive value="50"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **freq**

Super-types: [xs:string](#) < **freq** (by restriction)

Sub-types: None

Name	freq
Content	<ul style="list-style-type: none">Base XSD Type: string
Documentation	Frequency in real-world units, i.e. ".01hz" or "2kHz"

Schema Component Representation

```
<xs:simpleType name="freq">
  <xs:restriction base="xs:string"/>
</xs:simpleType>
```

Simple Type: **fx-bus**

Super-types: [xs:string](#) < **fx-bus** (by restriction)

Sub-types: None

Name	fx-bus
Content	<ul style="list-style-type: none">Base XSD Type: string<i>pattern</i> = FX\s?[1-6]

Schema Component Representation

```
<xs:simpleType name="fx-bus">
  <xs:restriction base="xs:string">
    <xs:pattern value="FX\s?[1-6]"/>
  </xs:restriction>
</xs:simpleType>
```

Simple Type: **glide-mode**

Super-types: [xs:string](#) < **glide-mode** (by restriction)

Sub-types: None

Name	glide-mode
Content	<ul style="list-style-type: none">Base XSD Type: string<i>value</i> comes from list: {'Held' 'All'}

Schema Component Representation

```
<xs:simpleType name="glide-mode">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Held"/>
    <xs:enumeration value="All"/>
  </xs:restriction>
</xs:simpleType>
```

Simple Type: **group-output**

Super-types: None

Sub-types: None

Name	group-output

Content

- Union of following types:
 - [output](#)
 - Locally defined type:
 - Base XSD Type: string
 - *value* comes from list: {'(ch)'}

Schema Component Representation

```
<xs:simpleType name="group-output">
  <xs:union memberTypes="output "
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="(ch)"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>
```

[top](#)

Simple Type: **interpolation**

Super-types: [xs:string](#) < **interpolation** (by restriction)

Sub-types: None

Name	interpolation
Content	<ul style="list-style-type: none">• Base XSD Type: string• <i>value</i> comes from list: {'Default' 'Draft' 'Hermite' 'Sinc32' 'Sinc64' 'Sinc128'}
Documentation	TXv2 only

Schema Component Representation

```
<xs:simpleType name="interpolation">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Default"/>
    <xs:enumeration value="Draft"/>
    <xs:enumeration value="Hermite"/>
    <xs:enumeration value="Sinc32"/>
    <xs:enumeration value="Sinc64"/>
    <xs:enumeration value="Sinc128"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **lfomode**

Super-types: [xs:string](#) < **lfomode** (by restriction)

Sub-types: None

Name	lfomode
------	---------

Content

- Base XSD Type: string
- *value* comes from list: {'Normal'|'Tempo'}

Schema Component Representation

```
<xs:simpleType name="Ifomode">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Normal"/>
    <xs:enumeration value="Tempo"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **lfotype**

Super-types: [xs:string](#) < **lfotype** (by restriction)

Sub-types: None

Name

lfotype

Content

- Base XSD Type: string
- *value* comes from list: {'Triangle'|'Sinus'|'Sawtooth'|'Square'|'Exponent'}

Schema Component Representation

```
<xs:simpleType name="lfotype">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Triangle"/>
    <xs:enumeration value="Sinus"/>
    <xs:enumeration value="Sawtooth"/>
    <xs:enumeration value="Square"/>
    <xs:enumeration value="Exponent"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **loopmode**

Super-types: [xs:string](#) < **loopmode** (by restriction)

Sub-types: None

Name

loopmode

Content

- Base XSD Type: string
- *value* comes from list: {'None'|'Forward'|'Backward'|'Bidirectional'|'Slice'|'PinnedSlice'}

Schema Component Representation

```

<xs:simpleType name="loopmode">
  <xs:restriction base="xs:string">
    <xs:enumeration value="None"/>
    <xs:enumeration value="Forward"/>
    <xs:enumeration value="Backward"/>
    <xs:enumeration value="Bidirectional"/>
    <xs:enumeration value="Slice"/>
    <xs:enumeration value="PinnedSlice"/>
  </xs:restriction>
</xs:simpleType>

```

[top](#)

Simple Type: **midicc**

Super-types: None

Sub-types: None

Name	midicc
Content	<ul style="list-style-type: none"> Union of following types: <ul style="list-style-type: none"> Locally defined type: <ul style="list-style-type: none"> Base XSD Type: string <i>pattern</i> = Midi CC ([0-9][1-9][0-9]1[0-1][0-9]12[0-7]) Locally defined type: <ul style="list-style-type: none"> Base XSD Type: integer 0 ≤ <i>value</i> ≤ 127 Locally defined type: <ul style="list-style-type: none"> Base XSD Type: string

Schema Component Representation

```

<xs:simpleType name="midicc">
  <xs:union>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="Midi CC ([0-9][1-9][0-9]1[0-1][0-9]12[0-7])"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="xs:integer">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="127"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="xs:string"/>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>

```

[top](#)

Simple Type: **midinote**

Super-types: [xs:string](#) < **midinote** (by restriction)

Sub-types: None

Name	midinote
Content	<ul style="list-style-type: none">Base XSD Type: string$pattern = [a-zA-Z]\#?-[0-9][0-9]?\d{1,3}$

Schema Component Representation

```
<xs:simpleType name="midinote">
  <xs:restriction base="xs:string">
    <xs:pattern value="[a-zA-Z]\#?-[0-9][0-9]?\d{1,3}"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **midivalue**

Super-types: [xs:integer](#) < **midivalue** (by restriction)

Sub-types: None

Name	midivalue
Content	<ul style="list-style-type: none">Base XSD Type: integer$0 \leq value \leq 127$

Schema Component Representation

```
<xs:simpleType name="midivalue">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="0"/>
    <xs:maxInclusive value="127"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **moddst**

Super-types: [xs:string](#) < **moddst** (by restriction)

Sub-types: None

Name	moddst
Content	<ul style="list-style-type: none">Base XSD Type: stringvalue comes from list: {'--' 'AEG Attack' 'AEG Time' 'AEG Amp' 'ENV1 Amp' 'ENV2 Amp' 'LFO1 Rate' 'LFO1 Amp' 'LFO2 Rate' 'LFO2 Amp' 'SEQ1 Rate' 'SEQ1 Amp' 'SEQ2 Rate' 'SEQ2 Amp' 'SEQ3 Rate' 'SEQ4

	Amp' Filter Freq' Filter Res' Filter Drive' Filter 1 Freq' Filter 1 Res' Filter 1 Drive' Filter 2 Freq' Filter 2 Res' Filter 2 Drive' Pitch' Pitch (raw)' Volume' Pan' Wave Start' Start' Delay' Loop Start' Loop End' Loop Direction' Glide' Insert Par 1' Insert Par 2' Send Level 1' Send Level 2' Send Level 3' Amp'}
Documentation	Accepted modulation destinations. See manual for details.

Schema Component Representation

```

<xs:simpleType name="moddst">
  <xs:restriction base="xs:string">
    <xs:enumeration value="--"/>
    <xs:enumeration value="AEG Attack"/>
    <xs:enumeration value="AEG Time"/>
    <xs:enumeration value="AEG Amp"/>
    <xs:enumeration value="ENV1 Amp"/>
    <xs:enumeration value="ENV2 Amp"/>
    <xs:enumeration value="LFO1 Rate"/>
    <xs:enumeration value="LFO1 Amp"/>
    <xs:enumeration value="LFO2 Rate"/>
    <xs:enumeration value="LFO2 Amp"/>
    <xs:enumeration value="SEQ1 Rate"/>
    <xs:enumeration value="SEQ1 Amp"/>
    <xs:enumeration value="SEQ2 Rate"/>
    <xs:enumeration value="SEQ2 Amp"/>
    <xs:enumeration value="SEQ3 Rate"/>
    <xs:enumeration value="SEQ4 Amp"/>
    <xs:enumeration value="Filter Freq"/>
    <xs:enumeration value="Filter Res"/>
    <xs:enumeration value="Filter Drive"/>
    <xs:enumeration value="Filter 1 Freq"/>
    <xs:enumeration value="Filter 1 Res"/>
    <xs:enumeration value="Filter 1 Drive"/>
    <xs:enumeration value="Filter 2 Freq"/>
    <xs:enumeration value="Filter 2 Res"/>
    <xs:enumeration value="Filter 2 Drive"/>
    <xs:enumeration value="Pitch"/>
    <xs:enumeration value="Pitch (raw)"/>
    <xs:enumeration value="Volume"/>
    <xs:enumeration value="Pan"/>
    <!-- legacy -->
    <xs:enumeration value="Wave Start"/>
    <xs:enumeration value="Start"/>
    <xs:enumeration value="Delay"/>
    <xs:enumeration value="Loop Start"/>
    <xs:enumeration value="Loop End"/>
    <xs:enumeration value="Loop Direction"/>
    <xs:enumeration value="Glide"/>
    <xs:enumeration value="Insert Par 1"/>
    <xs:enumeration value="Insert Par 2"/>
    <xs:enumeration value="Send Level 1"/>
    <xs:enumeration value="Send Level 2"/>
    <xs:enumeration value="Send Level 3"/>
    <xs:enumeration value="Amp"/>
  </xs:restriction>
</xs:simpleType>

```

Simple Type: **modsrc**

Super-types:	None
Sub-types:	None

Name	modsrc
Content	<ul style="list-style-type: none">Union of following types:<ul style="list-style-type: none">xctrlmidiccautomationLocally defined type:<ul style="list-style-type: none">Base XSD Type: stringvalue comes from list: {'--' 'AEG' 'ENV1' 'ENV2' 'LFO1' 'LFO2' 'SEQ1' 'SEQ2' 'SEQ3' 'Pitchbend' 'Mod Wheel' 'Pressure' 'Channel Pressure' 'Aftertouch' 'Random' 'Key' 'Key/R' 'Key/P' 'Vel' 'Vel/R' 'Vel/P' 'BPM/R'}
Documentation	Accepted modulation sources. See manual for details.

Schema Component Representation

```
<xs:simpleType name="modsrc">
  <xs:union memberTypes="xctrl midicc automation ">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="--"/>
        <xs:enumeration value="AEG"/>
        <xs:enumeration value="ENV1"/>
        <xs:enumeration value="ENV2"/>
        <xs:enumeration value="LFO1"/>
        <xs:enumeration value="LFO2"/>
        <xs:enumeration value="SEQ1"/>
        <xs:enumeration value="SEQ2"/>
        <xs:enumeration value="SEQ3"/>
        <xs:enumeration value="Pitchbend"/>
        <xs:enumeration value="Mod Wheel"/>
        <xs:enumeration value="Pressure"/>
        <xs:enumeration value="Channel Pressure"/>
        <xs:enumeration value="Aftertouch"/>
        <xs:enumeration value="Random"/>
        <xs:enumeration value="Key"/>
        <xs:enumeration value="Key/R"/>
        <xs:enumeration value="Key/P"/>
        <xs:enumeration value="Vel"/>
        <xs:enumeration value="Vel/R"/>
        <xs:enumeration value="Vel/P"/>
        <xs:enumeration value="BPM/R"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>
```

[top](#)

Simple Type: **mute**

--

Super-types: [xs:string](#) < **mute** (by restriction)

Sub-types: None

Name	mute
Content	<ul style="list-style-type: none">Base XSD Type: string<i>value</i> comes from list: {'0' 'false' '1' '2' 'by-solo' 'true'}

Schema Component Representation

```
<xs:simpleType name="mute">
  <xs:restriction base="xs:string">
    <xs:enumeration value="0"/>
    <xs:enumeration value="false"/>
    <!-- Muted -->
    <xs:enumeration value="1"/>
    <!-- Muted by solo -->
    <xs:enumeration value="2"/>
    <xs:enumeration value="by-solo"/>
    <!-- legacy, interpreted as '1' -->
    <xs:enumeration value="true"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **noteprio**

Super-types: [xs:string](#) < **noteprio** (by restriction)

Sub-types: None

Name	noteprio
Content	<ul style="list-style-type: none">Base XSD Type: string<i>value</i> comes from list: {'Last' 'Lowest' 'Highest'}
Documentation	Note priority, determines which key will be played when doing monophonic note stealing

Schema Component Representation

```
<xs:simpleType name="noteprio">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Last"/>
    <xs:enumeration value="Lowest"/>
    <xs:enumeration value="Highest"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **output**

Super-types: [xs:string](#) < **output** (by restriction)

Sub-types: None

Name	output
Content	<ul style="list-style-type: none">Base XSD Type: string

Schema Component Representation

```
<xs:simpleType name="output">
  <xs:restriction base="xs:string"/>
</xs:simpleType>
```

[top](#)

Simple Type: **pan**

Super-types: None

Sub-types: None

Name	pan
Content	<ul style="list-style-type: none">Union of following types:<ul style="list-style-type: none">Locally defined type:<ul style="list-style-type: none">Base XSD Type: float-1 <= value <= 1Locally defined type:<ul style="list-style-type: none">Base XSD Type: stringpattern = -(100\d{1,2}[\.,]?d{0,2})\s*%

Schema Component Representation

```
<xs:simpleType name="pan">
  <xs:union>
    <xs:simpleType>
      <xs:restriction base="xs:float">
        <xs:minInclusive value="-1"/>
        <xs:maxInclusive value="1"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="-(100\d{1,2}[\.,]?d{0,2})\s*%"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>
```

[top](#)

Simple Type: **percent**

Super-types: [xs:string](#) < **percent** (by restriction)

Sub-types: None

Name	percent
Content	<ul style="list-style-type: none">Base XSD Type: string<i>pattern</i> = (100\d{1,2})\s*%

Schema Component Representation

```
<xs:simpleType name="percent">
  <xs:restriction base="xs:string">
    <xs:pattern value="(100\d{1,2})\s*%" />
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **playback**

Super-types: [xs:string](#) < **playback** (by restriction)

Sub-types: None

Name	playback
Content	<ul style="list-style-type: none">Base XSD Type: string<i>value</i> comes from list: {'Resample' 'Pitchshift' 'Timestretch'}

Schema Component Representation

```
<xs:simpleType name="playback">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Resample" />
    <xs:enumeration value="Pitchshift" />
    <xs:enumeration value="Timestretch" />
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **playmode**

Super-types: [xs:string](#) < **playmode** (by restriction)

Sub-types: None

Name	playmode
Content	<ul style="list-style-type: none">Base XSD Type: string<i>value</i> comes from list: {'Normal' 'Oneshot' 'Toggle' 'Release'}

Schema Component Representation

```

<xs:simpleType name="playmode">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Normal"/>
    <xs:enumeration value="Oneshot"/>
    <xs:enumeration value="Toggle"/>
    <xs:enumeration value="Release"/>
  </xs:restriction>
</xs:simpleType>

```

[top](#)

Simple Type: **pluginversion**

Super-types: [xs:integer](#) < **pluginversion** (by restriction)

Sub-types: None

Name	pluginversion
Content	<ul style="list-style-type: none"> Base XSD Type: integer
Documentation	Version of creating plugin. Version 1.0.0 was 1000, v1.0.1.1 was 1011

Schema Component Representation

```

<xs:simpleType name="pluginversion">
  <xs:restriction base="xs:integer"/>
</xs:simpleType>

```

[top](#)

Simple Type: **polar-percent**

Super-types: [xs:string](#) < **polar-percent** (by restriction)

Sub-types: None

Name	polar-percent
Content	<ul style="list-style-type: none"> Base XSD Type: string <i>pattern</i> = <code>-(?(100 d{1,2})\s*%)</code>

Schema Component Representation

```

<xs:simpleType name="polar-percent">
  <xs:restriction base="xs:string">
    <xs:pattern value="-(?(100|d{1,2})\s*%)" />
  </xs:restriction>
</xs:simpleType>

```

[top](#)

Simple Type: **poly**

Super-types: [xs:integer](#) < **poly** (by restriction)

Sub-types: None

Name	poly
Content	<ul style="list-style-type: none">Base XSD Type: integer1 <= <i>value</i> <= 256

Schema Component Representation

```
<xs:simpleType name="poly">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="1"/>
    <xs:maxInclusive value="256"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **polymode**

Super-types: [xs:string](#) < **polymode** (by restriction)

Sub-types: None

Name	polymode
Content	<ul style="list-style-type: none">Base XSD Type: string<i>value</i> comes from list: {'Poly' 'Mono' 'Legato' 'Mono/P' 'Legato/P'}

Schema Component Representation

```
<xs:simpleType name="polymode">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Poly"/>
    <xs:enumeration value="Mono"/>
    <xs:enumeration value="Legato"/>
    <xs:enumeration value="Mono/P"/>
    <xs:enumeration value="Legato/P"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **program-change-mode**

Super-types: [xs:string](#) < **program-change-mode** (by restriction)

Sub-types: None

Name	program-change-mode
Content	<ul style="list-style-type: none">Base XSD Type: string

- *value* comes from list: {'Performance'|'Program'|'Off'}

Schema Component Representation

```
<xs:simpleType name="program-change-mode">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Performance"/>
    <xs:enumeration value="Program"/>
    <xs:enumeration value="Off"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **quality**

Super-types: [xs:string](#) < **quality** (by restriction)

Sub-types: None

Name	quality
Content	<ul style="list-style-type: none">• Base XSD Type: string• <i>value</i> comes from list: {'Default' 'Draft' 'Normal' 'High' 'Higher' 'Offline 1' 'Offline 2'}
Documentation	TXv3 interpolation/algorithm quality

Schema Component Representation

```
<xs:simpleType name="quality">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Default"/>
    <xs:enumeration value="Draft"/>
    <xs:enumeration value="Normal"/>
    <xs:enumeration value="High"/>
    <xs:enumeration value="Higher"/>
    <xs:enumeration value="Offline 1"/>
    <xs:enumeration value="Offline 2"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **rootkey**

Super-types: [xs:string](#) < **rootkey** (by restriction)

Sub-types: None

Name	rootkey
Content	<ul style="list-style-type: none">• Base XSD Type: string• <i>pattern</i> = [a-zA-Z]#?-[0-9][0-9]? \d{1,3} no

Schema Component Representation

```
<xs:simpleType name="rootkey">
  <xs:restriction base="xs:string">
    <xs:pattern value="[a-zA-Z]#?-?[0-9][0-9]?|d{1,3}|no|"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **round-robin-source**

Super-types: [xs:string](#) < **round-robin-source** (by restriction)

Sub-types: None

Name	round-robin-source
Content	<ul style="list-style-type: none">Base XSD Type: string<i>value</i> comes from list: {'Default' 'Group' 'Program' 'MIDI'}

Schema Component Representation

```
<xs:simpleType name="round-robin-source">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Default"/>
    <xs:enumeration value="Group"/>
    <xs:enumeration value="Program"/>
    <xs:enumeration value="MIDI"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **scale**

Super-types: [xs:integer](#) < **scale** (by restriction)

Sub-types: None

Name	scale
Content	<ul style="list-style-type: none">Base XSD Type: integer$-1200 \leq \textit{value} \leq 1200$

Schema Component Representation

```
<xs:simpleType name="scale">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="-1200"/>
    <xs:maxInclusive value="1200"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **send-destination**

Super-types: None

Sub-types: None

Name	send-destination
Content	<ul style="list-style-type: none">Union of following types:<ul style="list-style-type: none">fx-busoutputLocally defined type:<ul style="list-style-type: none">Base XSD Type: stringvalue comes from list: {'None'}

Schema Component Representation

```
<xs:simpleType name="send-destination">
  <xs:union memberTypes="fx-bus output ">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="None"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>
```

[top](#)

Simple Type: **seqlength**

Super-types: [xs:integer](#) < **seqlength** (by restriction)

Sub-types: None

Name	seqlength
Content	<ul style="list-style-type: none">Base XSD Type: integervalue comes from list: {'0' '8' '16' '32' '64' '128'}

Schema Component Representation

```
<xs:simpleType name="seqlength">
  <xs:restriction base="xs:integer">
    <xs:enumeration value="0"/>
    <xs:enumeration value="8"/>
    <xs:enumeration value="16"/>
    <xs:enumeration value="32"/>
    <xs:enumeration value="64"/>
    <xs:enumeration value="128"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **sequence**

Super-types: [xs:string](#) < **sequence** (by restriction)

Sub-types: None

Name	sequence
Content	<ul style="list-style-type: none">Base XSD Type: string<i>pattern</i> = <code>(-?\d{1,3},?)* </code>
Documentation	Step seq values, comma separated list of -1<->1 values. See "seqlength" for valid list lengths. If elements are missing or overflow, they are silently ignored.

Schema Component Representation

```
<xs:simpleType name="sequence">  
  <xs:restriction base="xs:string">  
    <xs:pattern value="(-?\d{1,3},?)*|"/>  
  </xs:restriction>  
</xs:simpleType>
```

[top](#)

Simple Type: **solo**

Super-types: [xs:boolean](#) < **solo** (by restriction)

Sub-types: None

Name	solo
Content	<ul style="list-style-type: none">Base XSD Type: boolean

Schema Component Representation

```
<xs:simpleType name="solo">  
  <xs:restriction base="xs:boolean"/>  
</xs:simpleType>
```

[top](#)

Simple Type: **synctype**

Super-types: [xs:string](#) < **synctype** (by restriction)

Sub-types: None

Name	synctype
Content	<ul style="list-style-type: none">Base XSD Type: string<i>value</i> comes from list: {'None' 'Key' 'Group' 'Voice' 'Tempo'}

Schema Component Representation

```
<xs:simpleType name="synctype">
  <xs:restriction base="xs:string">
    <xs:enumeration value="None"/>
    <xs:enumeration value="Key"/>
    <xs:enumeration value="Group"/>
    <xs:enumeration value="Voice"/>
    <xs:enumeration value="Tempo"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **tempo**

Super-types: None

Sub-types: None

Name	tempo
Content	<ul style="list-style-type: none">Union of following types:<ul style="list-style-type: none">Locally defined type:<ul style="list-style-type: none">Base XSD Type: floatLocally defined type:<ul style="list-style-type: none">Base XSD Type: string<i>pattern</i> = \d*\.\?d*s*(BPM bpm)

Schema Component Representation

```
<xs:simpleType name="tempo">
  <xs:union>
    <xs:simpleType>
      <xs:restriction base="xs:float"/>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="\d*\.\?d*s*(BPM|bpm)"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>
```

[top](#)

Simple Type: **time**

Super-types: [xs:string](#) < **time** (by restriction)

Sub-types: None

Name	time
Content	<ul style="list-style-type: none">Base XSD Type: string
Documentation	Time in real-world unit. I.e. "12ms" or "1s" or "1000ms". Range is . 1ms -> ~38s

Schema Component Representation

```
<xs:simpleType name="time">
  <xs:restriction base="xs:string"/>
</xs:simpleType>
```

[top](#)

Simple Type: **trigtype**

Super-types: [xs:string](#) < **trigtype** (by restriction)

Sub-types: None

Name	trigtype
Content	<ul style="list-style-type: none">Base XSD Type: stringvalue comes from list: {'Retrig' 'Continuous'}
Documentation	legacy - not used past tx 2.2.4

Schema Component Representation

```
<xs:simpleType name="trigtype">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Retrig"/>
    <xs:enumeration value="Continuous"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **tuning**

Super-types: [xs:string](#) < **tuning** (by restriction)

Sub-types: None

Name	tuning
Content	<ul style="list-style-type: none">Base XSD Type: stringvalue comes from list: {'Inv * 4' 'Inv * 2' 'Inv' 'Inv 1/2' 'Inv 1/4' 'Fixed' '1/4' '1/2' 'Normal' '2' '4'}

Schema Component Representation

```
<xs:simpleType name="tuning">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Inv * 4"/>
    <xs:enumeration value="Inv * 2"/>
    <xs:enumeration value="Inv"/>
    <xs:enumeration value="Inv 1/2"/>
    <xs:enumeration value="Inv 1/4"/>
    <xs:enumeration value="Fixed"/>
    <xs:enumeration value="1/4"/>
  </xs:restriction>
</xs:simpleType>
```

```

    <xs:enumeration value="1/2"/>
    <xs:enumeration value="Normal"/>
    <xs:enumeration value="* 2"/>
    <xs:enumeration value="* 4"/>
  </xs:restriction>
</xs:simpleType>

```

[top](#)

Simple Type: **uuid**

Super-types: [xs:string](#) < **uuid** (by restriction)

Sub-types: None

Name	uuid
Content	<ul style="list-style-type: none"> Base XSD Type: string <i>pattern</i> = [a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}

Schema Component Representation

```

<xs:simpleType name="uuid">
  <xs:restriction base="xs:string">
    <xs:pattern value="[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}"/>
  </xs:restriction>
</xs:simpleType>

```

[top](#)

Simple Type: **velocitycurve**

Super-types: [xs:string](#) < **velocitycurve** (by restriction)

Sub-types: None

Name	velocitycurve
Content	<ul style="list-style-type: none"> Base XSD Type: string <i>value</i> comes from list: {'Default' 'Linear' 'Exp1' 'Exp2' 'Exp3' 'Log1' 'Log2' 'Log3'}
Documentation	TXv2 only

Schema Component Representation

```

<xs:simpleType name="velocitycurve">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Default"/>
    <xs:enumeration value="Linear"/>
    <xs:enumeration value="Exp1"/>
    <xs:enumeration value="Exp2"/>
    <xs:enumeration value="Exp3"/>
  </xs:restriction>
</xs:simpleType>

```

```

    <xs:enumeration value="Log1"/>
    <xs:enumeration value="Log2"/>
    <xs:enumeration value="Log3"/>
  </xs:restriction>
</xs:simpleType>

```

[top](#)

Simple Type: **vol**

Super-types: None

Sub-types: None

Name	vol
Content	<ul style="list-style-type: none"> Union of following types: <ul style="list-style-type: none"> Locally defined type: <ul style="list-style-type: none"> Base XSD Type: float $0 \leq \text{value} \leq 1$ Locally defined type: <ul style="list-style-type: none"> Base XSD Type: string <i>pattern</i> = <code>-(\d*\.\d* inf)\s*dB</code> Locally defined type: <ul style="list-style-type: none"> Base XSD Type: string <i>pattern</i> = <code>-(100 \d{1,2}\.\d{0,2})\s*%</code>

Schema Component Representation

```

<xs:simpleType name="vol">
  <xs:union>
    <xs:simpleType>
      <xs:restriction base="xs:float">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="1"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="-(\d*\.\d*|inf)\s*dB"/>
      </xs:restriction>
    </xs:simpleType>
    <!-- ta bort -->
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="-(100|\d{1,2}\.\d{0,2})\s*%"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>

```

[top](#)

Simple Type: **waveref**

Super-types: [xs:string](#) < **waveref** (by restriction)

Sub-types: None

Name	waveref
Content	<ul style="list-style-type: none"> Base XSD Type: string <i>pattern</i> = <none> (\d+)

Schema Component Representation

```
<xs:simpleType name="waveref">
  <xs:restriction base="xs:string">
    <xs:pattern value="<none>|(\d+)"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)

Simple Type: **xctrl**

Super-types: [xs:string](#) < **xctrl** (by restriction)

Sub-types: None

Name	xctrl
Content	<ul style="list-style-type: none"> Base XSD Type: string <i>pattern</i> = External Controller ([1-9]1[0-6])

Schema Component Representation

```
<xs:simpleType name="xctrl">
  <xs:restriction base="xs:string">
    <xs:pattern value="External Controller ([1-9]1[0-6])"/>
  </xs:restriction>
</xs:simpleType>
```

[top](#)