DipTrace Schematic Capture — XML Format Specification

Generated from the DipTrace serializer, repository revision 7276 — reflects current program behaviour.

Contents

How to read this specification 1
Document structure 1
Value types & formatting 2
Coordinate frames, units & angles 2
Identifiers, references & list ordering 2
Object state flags 3
Plug-in exchange & import merge 3
Cross-cutting: font-size fields 4
<SheetSettings> — page setup and title blocks 4
<Settings> — global project settings 6
<NetClasses> / <NetClass> 7
<DesignCache> 8
<ERC> 9
<Components> / <Part> 9
<Nets> / <Net> 11
<DifferentialPairs> / <DifferentialPair> 13
<Buses> / <Bus> and <BusConnectors> / <BusConnector> 13
<Shapes> / <Shape> 14
<Tables> / <Table> / <Cell> 16
<Simulator> — SPICE setup 18
<Groups> / <Group> 19

How to read this specification

This document describes the **DipTrace Schematic Capture** flavour of the DipTrace XML format — the same format used both for `File ▸ Save/Open As ▸ DipTrace XML` and for the plug-in exchange file. It is generated directly from the DipTrace serializer and reflects what the program actually reads and writes.

Each element is given with a short description, a representative XML fragment, and a table of its attributes. The **Written** column states the condition under which the serializer emits the attribute (`always`, a guard such as `if >-1`, or a context such as `Text shapes only`); an attribute left at its default is usually omitted, so a missing attribute means *use the default*, not zero.

Document structure

A schematic project is a single `<Source>` root holding the embedded design-cache library and the editable schematic:

```
<Source Type="DipTrace-Schematic" Version="4.3.0.x" Units="mm">
  <Library Type="DipTrace-ComponentLibrary" ...>...</Library>   <!-- design cache
-->
  <Schematic> ... </Schematic>                                  <!-- the editable
project -->
</Source>
```

The editable schematic data lives under `<Schematic>` (a plug-in's data node is `/Source/Schematic`); the sibling `<Library>` is the project's design cache and nests the footprint (pattern) library.

Value types & formatting

Type	Encoding
`Int`	A plain decimal integer.
`Real`	A decimal number expressed in the file's `Units`, written with a **dot** decimal separator. The reader is tolerant (it normalises both dot and comma), but the dot is canonical and is what other tools expect.
`Bool`	`Y` / `N`. (A few Schematic simulator fields use `+` / `-`.)
`Text`	A string attribute or element text.
`Color`	A 24-bit integer in Windows `0x00BBGGRR` order (e.g. `255` = red).
enum	A fixed set of text (or, in a few cases, integer) values, listed with the element.

Coordinate frames, units & angles

- The document root carries `Units="mm" | "inch" | "mil"`. **Every** `Real` coordinate, length and size in the file is in those units; there is no per-object override.
- Objects inside a library symbol or footprint (pins, pads, shapes) are stored in an **object-centre** frame — coordinates are relative to the part/footprint origin and are not affected by where an instance is placed.
- Angles are in **radians, counter-clockwise**, unless an attribute is explicitly a discrete orientation enum (e.g. a footprint `Orientation` of `0/90/180/270`).
- Coordinate scaling and sign are internal to the file; read and write values as they are — do not rescale or flip them.

Identifiers, references & list ordering

- Most list elements carry an `Id` (or `Index`/`Number`) that other objects reference. A reference value of `-1` means *none / not connected*.
- A few lists are referenced **positionally**, not by an `Id` attribute: a `ViaStyle` value indexes the in-order `<ViaStyles>` list and a `NetClass` value indexes the in-order `<NetClasses>` list. Reordering those lists silently reassigns the references.
- When a whole file is opened, top-level object lists are resolved **by position**: each array must be dense, ascending from `Id` 0, with position equal to `Id`. A file authored out of order or with gaps will bind references to the wrong objects.

Object state flags

Attribute	Type	Meaning
`Selected`	Bool	User selection state.
`Locked`	Bool	Edit lock.
`Group`	Int	Group `Id`, or `-1` for none; groups are listed in a `<Groups>` section.
`Enabled`	Bool	*Exists / removed.* A normal save prunes disabled objects, so a saved file rarely shows `Enabled="N"`; on the plug-in exchange / edit-import path, however, `Enabled="N"` is honoured as a **delete** flag on the editable objects. An absent flag means enabled.

Plug-in exchange & import merge

The same format is the **plug-in exchange** file: DipTrace exports the selected data, launches the plug-in with the file path as its argument, waits for it to exit, then re-imports the (possibly edited) file. How the result merges back depends on the import mode:

- **Whole-list (`ImpMode=All`)** — the incoming list *replaces* the project's list of that object kind; anything omitted is removed.
- **`Edit`** — each **top-level** object is matched to an existing one **by its `Id`** and overwritten in place; an object with a new or absent `Id` is added. Keep an object's `Id` to edit it, and **omit the `Id`** on objects you add — never invent one, because over a partial export a computed "free" Id can collide with an unexported object and overwrite it.

> **Nested lists without their own `Id` are REPLACED, not merged.** A net's `<Traces>`, a net's or bus's `<Wires>`, and point lists have no per-member `Id`. When such a container is present in your XML, DipTrace **discards the existing list and rebuilds it from exactly the children you supply** — so appending one trace under an existing net replaces that net's *entire* routing. To add one member, re-list every current member plus the new one; to leave the list untouched, omit the container element entirely.

> **The `Selected` filter skips — it does not add.** When a plug-in runs with a `Selected` filter under `Edit`, only objects flagged `Selected="Y"` are processed at all: a marked object with an existing `Id` is edited, a marked object with a new `Id` is added, and an **unflagged** object is **skipped entirely** (neither edited nor added). Put `Selected="Y"` on every object you add or edit under such a filter, or its data is silently dropped.

To remove an object, set `Enabled="N"` on it and leave it in the file (see *Object state flags*) — do not delete the XML node. Component and Pattern *library* files use coarser Library / Component / Part import modes rather than per-object `Edit`, but the `Enabled` delete flag and the `Id` rules still apply.

A note on completeness. Where a newer serializer field has no counterpart in an older published table (for example the real-valued/mono font fields, or per-object shield-group ids), it is documented here as a first-class attribute. Preserve attributes and elements you do not recognise when editing a file in place — they belong to other subsystems and to future format versions.

Cross-cutting: font-size fields

Every text-bearing element in the Schematic dialect carries a legacy rounded integer font size *and* a fractional companion. On import the fractional value overrides the rounded integer whenever it is present, so a reader that wants exact sizes must prefer the `…Float` attribute.

Attribute	Type	Written	Description
`FontSize` / `Font` / `FontSizeFloat`	Int / Real	always (as a pair)	The integer is the rounded legacy size; `FontSizeFloat` holds the true fractional size and wins on import. Written on `<BorderZones>`, `<Field>`, `<Markings>`, `<BusConnections>`, `<PinNumbers>`, `<NetName>`, a text `<Shape>`, `<Table>`, and `<Cell>`.
`MarkingFontSize` / `MarkingFontSizeFloat`	Int / Real	always (as a pair)	Same pairing for a `<Part>`'s marking font; the `…Float` twin carries the fractional value.
`FontMono`	Bool (Y/N)	vector font only	Written only when the object uses the internal vector font (`FontVector="Y"`); selects the monospaced vector face. Absent for TrueType text.
`TextWidth` / `TextHeight`	Real	export-only	Computed bounding size of the text (via `GetTextWH`), emitted for informational use on `<Field>`, text `<Shape>`, and `<Cell>`. Recomputed on import — never read back; treat as read-only.

Note. The fractional/mono font family is written throughout the document but does not replace the integer fields — both are always present so older readers keep working. When authoring XML, set the `…Float` value; the rounded integer is derived.

<SheetSettings> — page setup and title blocks

Holds the sheet list and per-sheet page geometry, border zones, and up to six title blocks. Objects across the schematic reference a sheet by its `Id`.

```
<SheetSettings>
  <DisplayTitles>Y</DisplayTitles>
  <DisplaySheet>Y</DisplaySheet>
  <ActiveSheet>0</ActiveSheet>
  <Sheets>
    <Sheet>
      <Id>0</Id> <Name>Sheet1</Name> <Type>Normal</Type>
      <XPos>0</XPos> <YPos>0</YPos> <Scale>1</Scale>
      <SheetWidth>297</SheetWidth> <SheetHeight>210</SheetHeight>
      <LeftMargin>5</LeftMargin> <TopMargin>5</TopMargin>
      <RightMargin>5</RightMargin> <BottomMargin>5</BottomMargin>
      <BorderZones>…</BorderZones>
    </Sheet>
  </Sheets>
</SheetSettings>
```

Header children: `DisplayTitles` and `DisplaySheet` (Bool) toggle title-block / sheet-border display; `ActiveSheet` (Int) is the currently shown sheet index.

<Sheet>

Attribute	Type	Written	Description
`Id`	Int	always	Sheet index; objects reference it by `Sheet="N"`.
`Name`	Text	always	Sheet name.
`Type`	enum	always	`0`=Normal, `1`=Hierarchy Block.
`BlockId`	Int	if Type is Hierarchy Block	Links this child sheet to the hierarchy-block *part* that expands into it (see the note below); omitted entirely for a normal sheet.
`XPos` / `YPos`	Real	always	Sheet origin (Y is negated on write).
`Scale`	Real	always	Display scale.
`SheetWidth` / `SheetHeight`	Real	always	Page size in file Units.
`LeftMargin` / `TopMargin` / `RightMargin` / `BottomMargin`	Real	always	Printable margins.

Hierarchy blocks. A hierarchical design links a *block part* (a `<Part>` of part type 4, carrying a `BlockId`) to the child `<Sheet>` that expands it. The link is the matching `BlockId` value written on both sides — a content link that is **independent of the positional `Sheet Id`** that ordinary objects use through `Sheet="N"`. Do not conflate the two: a sheet's `Id` is its page

index, its `BlockId` is its block identity. If a `BlockId` does not resolve on import, DipTrace falls back to matching the block to its sheet by name.

<BorderZones> and the six title blocks

Written only in a full save (not per-object plug-in exports). `<BorderZones>` carries the zone-grid setup: `Visible`, `HorzZones`/`VertZones` (Int), `Standard`, `FontName`, `FontSize`/`FontSizeFloat`, `FontLineWidth` (Real), `Border` (Bool), and `HorzBorderSize`/`VertBorderSize` (Real). The six title blocks are separate elements: `<BottomRightBlock>`, `<BottomLeftBlock>`, `<TopRightBlock>`, `<TopLeftBlock>`, `<ExtTopLeftBlock>`, `<ExtBottomLeftBlock>`.

```
<BottomRightBlock Width="180" Height="55">
  <Name>ANSI A</Name>
  <ColumnWidths><Item>40</Item>…</ColumnWidths>
  <RowHeights><Item>10</Item>…</RowHeights>
  <Cells><Item><Item FieldId="0" TopLine="Y" BottomLine="Y" LeftLine="Y"
RightLine="Y"/>…</Item></Cells>
  <Fields>
    <Field Id="0" FontVector="N" FontSize="10" FontSizeFloat="10" FontWidth="-2"
           FontScale="1" FontColor="0" LineSpacing="1.2" TextShow="Text"
TextAlign="Left"
           X1="0" Y1="0" X2="40" Y2="10" TextWidth="18" TextHeight="4">
      <TextLines><TextLine>Title</TextLine></TextLines>
      <FontName>Tahoma</FontName>
    </Field>
  </Fields>
</BottomRightBlock>
```

Note. A title block is omitted when its `Name` is blank, and `ColumnWidths`/`RowHeights`/`Cells`/`Fields` are omitted when empty — all optional. A `<Field>` emits `FontMono` only when `FontVector="Y"`, and `TextWidth`/`TextHeight` only for a plain text field (`TextShow="Text"`).

<Settings> — global project settings

Global fonts, grid, origin, line widths, bus-connection display, pin numbers, and the net-label font. Emitted as a block of child elements.

```
<Settings>
  <Markings FontVector="Y" FontMono="N" FontSize="10" FontSizeFloat="10"
            FontWidth="-2" FontScale="1" UsePartFontColor="N" FontColor="0">
    <RefDesGlobal Show="Common" ShowPart="Common" Align="Auto"/>
    <NameGlobal …/> <ValueGlobal …/> <ManufacturerGlobal …/> <DatasheetGlobal …/>
  </Markings>
  <Grid Visible="Y" Snap="Y" Size="2.5"/>
  <Origin Visible="N" AxisColor="0" X="0" Y="0"/>
  <LineWidth Wire="0.25" NodeSize="1" Bus="0.5" Table="0.25" Titles="0.25"/>
  <BusConnections Show="NetName" Position="Above" FontSize="8" FontSizeFloat="8"/
>
  <PinNumbers Visible="Y" Font="8" FontSizeFloat="8"/>
```

```
  <NetName FontSize="8" FontSizeFloat="8"/>
  <HidePower>N</HidePower>
  <ProjectDir>…</ProjectDir>
</Settings>
```

`<Markings>` holds the shared component-marking font (`FontVector`, `FontMono` in the vector branch, `FontName` in the TrueType branch, plus `FontSize`/`FontSizeFloat`/`FontWidth`/`FontScale`/`UsePartFontColor`/`FontColor`) and per-field show/align children `<RefDesGlobal>`, `<NameGlobal>`, `<ValueGlobal>`, `<ManufacturerGlobal>`, `<DatasheetGlobal>`, and an optional `<AddFieldsGlobal>` list.

Element / Attr	Type	Written	Description
`<Grid>` `Visible`/`Snap`/`Size`	Bool/ Bool/ Real	always	Grid display, snapping, spacing.
`<Origin>` `X`/`Y`/`Visible`/`AxisColor`	Real/ Real/ Bool/ Int	always	User origin position, visibility, axis color.
`<LineWidth>` `Wire`/`NodeSize`/`Bus`/`Table`/`Titles`	Real	always	Default widths for wires, junction node size, buses, table lines, title lines.
`<BusConnections>` `Show`/ `Position`/`FontSize`/ `FontSizeFloat`	enum/ enum/ Int/ Real	always	`Show`: NetName/Number; `Position`: Above/Below; the label font size.
`<PinNumbers>` `Visible`/`Font`/ `FontSizeFloat`	Bool/ Int/ Real	always	Pin-number display and font size.
`<NetName>` `FontSize`/ `FontSizeFloat`	Int/ Real	always	Net-label font size (schematic net names), between `<PinNumbers>` and `<HidePower>`.
`<HidePower>`	Bool	always	Hide power/ground pins.
`<ProjectDir>`	Text	always	Project directory.

<NetClasses> / <NetClass>

The project's net classes, written in order. The diff-pair rule attributes appear only for a Differential Pair class.

```
<NetClasses>
  <NetClass Id="0" UpdateId="0" Type="Normal" Width="0.25" Clearance="0.25">
    <Name>Default</Name>
```

```
  </NetClass>
  <NetClass Id="1" UpdateId="3" Type="Differential Pair" Width="0.2"
Clearance="0.2"
            DifClearance="0.2" MaxUncoupledLength="5" Tolerance="0.1" Phase="0"
Phase_ErrorLength="0">
    <Name>DiffPair</Name>
  </NetClass>
</NetClasses>
```

Attribute	Type	Written	Description
`Id`	Int	always	In-order index of the class (equals its list position).
`UpdateId`	Int	always	Cross-link id for update against the PCB.
`Type`	enum	always	`0`=Normal, `1`=Differential Pair.
`Width`	Real	if >−1	Default trace width.
`Clearance`	Real	if >−1	Default clearance.
`DifClearance`	Real	Diff Pair, if >−1	In-pair clearance.
`MaxUncoupledLength`	Real	Diff Pair only	Max uncoupled length.
`Tolerance`	Real	Diff Pair only	Length-match tolerance.
`Phase`	Real	Diff Pair only	Phase-match target.
`Phase_ErrorLength`	Real	Diff Pair only	Phase error length.
`<Name>`	Text	always	Class name (child element).

Note. A net selects its class with `NetClass="N"`, a **positional index** into this in-order list. On a native save `Id` equals that position, so the two agree; reordering `<NetClass>` entries silently reassigns classes.

<DesignCache>

Records how many leading components in the embedded `<Library>` are design-cache entries.

```
<DesignCache Count="3"/>
```

Attribute	Type	Written	Description
`Count`	Int	always	Number of leading design-cache components (stored as the internal cache count plus one).

<ERC>

Electrical-rules-check flags and the power/ground name templates.

```
<ERC CheckPinType="Y" CheckNotConnected="Y" CheckSinglePin="Y" CheckShort="Y"
CheckPinSuperimpose="Y">
  <VCCTemplate>VCC;+*</VCCTemplate>
  <GNDTemplate>GND;0</GNDTemplate>
</ERC>
```

Attribute	Type	Written	Description
`CheckPinType`	Bool	always	Check pin-type conflicts.
`CheckNotConnected`	Bool	always	Check not-connected pins.
`CheckSinglePin`	Bool	always	Check single-pin nets.
`CheckShort`	Bool	always	Check shorts.
`CheckPinSuperimpose`	Bool	always	Check superimposed pins.
`<VCCTemplate>` / `<GNDTemplate>`	Text	always	Power / ground net-name templates (child elements).

<Components> / <Part>

One `<Part>` per placed part/section; several parts of a multi-part component share a `RefDes`. `ComponentStyle` points to a symbol in the embedded library and `ComponentPart` selects the section within it.

```
<Part Id="0" UpdateId="2221" ComponentStyle="CompType229" ComponentPart="0"
AllowParts="N"
      PartNumber="1" Sheet="4" X="83.82" Y="-43.815" Angle="4.7124"
      MarkingFontSize="10" MarkingFontSizeFloat="10">
  <RefDes>C1</RefDes> <PartRefDes>1</PartRefDes> <PartName>Part 1</PartName>
  <Value>(1608)</Value> <Name>CAP_1608(0603)</Name> <BaseName>CAP</BaseName>
  <Manufacturer>Yageo</Manufacturer> <Datasheet>http://…</Datasheet>
  <LibPath><Path>…</Path><Var>…</Var></LibPath>
  <RefDesMarking Show="Common" Align="Auto" Horz="Left" Vert="Top" X="0" Y="0"
Angle="0"/>
  <NameMarking …/> <ValueMarking …/> <ManufacturerMarking …/> <DatasheetMarking
…/>
  <Pins><Pin NetId="0"/><Pin NetId="-1" NotConnected="Y"/></Pins>
</Part>
```

Attribute	Type	Written	Description
`Id`	Int	always	Part id; net pin items reference it.
`UpdateId`	Int	always	Cross-link id to the PCB component.

Attribute	Type	Written	Description
BlockId	Int	hierarchy-block part only	Hierarchy-block link (part type 4); matches the BlockId of the child <Sheet> it expands into.
ComponentStyle	Text	if library entry resolved	Symbol name in the embedded library.
ComponentPart	Int	always	Section index within that symbol.
AllowParts	Bool	always	Whether the component is split into parts/sections.
PartNumber	Int	always	This part's number within the component.
Sheet	Int	always	Owning sheet Id.
X / Y	Real	always	Placement (Y negated on write).
Angle	Real	if ≠0	Rotation, radians CCW.
HorzFlip / VertFlip	Bool	if flipped	Mirror flags (only Y is written).
ShowNumbers	enum	if >0	0=Common, 1=Show, 2=Hide.
ShowOrigin	Bool	if set	Show the part origin.
CustomMarkingFont	Bool	if set	Part overrides the global marking font.
MarkingFontSize / MarkingFontSizeFloat	Int / Real	always	Marking font size (float twin wins on import).
MarkingFontAll	Bool	if set	Apply the marking font to all fields.
Group	Int	if >−1	Group id, else omitted.
Selected / Locked	Bool	if set	Editor flags (only Y written).

Value elements (child text, all always written): <RefDes>, <PartRefDes>, <PartName>, <Value>, <Name>, <BaseName> (the component base name), <Manufacturer> (the manufacturer value, distinct from the display-only <ManufacturerMarking>), and <Datasheet> (the datasheet value, distinct from <DatasheetMarking>). Followed by <LibPath>, the five marking-display blocks (<RefDesMarking> … <DatasheetMarking>, each with Show/ShowPart/Align/Horz/Vert/X/Y/Angle), an optional <AddFields> list, the <Pins> list, and an optional <Category>.

<Pins> (of a Part)

The part's pins in symbol pin order; nets reference a pin by this positional index.

Attribute	Type	Written	Description
NetId	Int	always	Connected net Id, or -1 for none.

Attribute	Type	Written	Description
NotConnected	Bool	if set	Explicitly marked not-connected (only Y written).

Note. A normal save never writes Enabled on a <Part> — disabled parts are pruned. Since rev 7272 the plug-in importer honors Enabled="N" on a <Part> as a delete flag (see the closing note under <Groups>).

<Nets> / <Net>

Each net lists its pin memberships and its wires. Boolean flags are written only when set (as Y), except Enabled, which is written as N for a deleted net.

```
<Net Id="0" NetClass="0" HiddenId="0" Global="Y" CustomColor="Y" WireColor="255">
  <Name>AP-WAKE-BT</Name>
  <Pins><Item Part="596" Pin="0"/><Item Part="347" Pin="2"/></Pins>
  <Wires>
    <Wire Id="0" Sheet="1" Connected1="Pin" Bus1="-1" Object1="709"
SubObject1="0"
          Connected2="Wire" Bus2="-1" Object2="3" SubObject2="1"
          HiddenPower="N" CanUnhide="N">
      <Points><Point X="-1.27" Y="-13.97" Dir="-1"/><Point X="0" Y="-13.97"
Dir="0"/></Points>
    </Wire>
  </Wires>
</Net>
```

Attribute	Type	Written	Description
Id	Int	always	Net id; referenced by pins, shapes, buses, diff pairs.
NetClass	Int	always	Positional index into <NetClasses>.
HiddenId	Int	always	Internal stable id used across merge/update.
UniteByName	Bool	if set	Unite same-named nets.
ConnectPinByName	Bool	if set	Connect pins by name.
Global	Bool	if set	Global net.
CustomColor	Bool	if set	Use a custom wire color; when set, WireColor follows.
WireColor	Int	if custom & ≠0	Custom wire color.
Locked	Bool	if set	Locked.

Attribute	Type	Written	Description
`Enabled`	Bool	`N` only, for a deleted net	The one object that genuinely emits `Enabled="N"` (delete flag). Never written when enabled.
`<Name>`	Text	always	Net name (child element).

`<Pins>` holds `<Item Part="…" Pin="…"/>` — the part `Id` plus the pin's positional index in that part.

<Wire> / <Point>

Attribute	Type	Written	Description
`Id`	Int	always	Wire id within the net.
`Sheet`	Int	always	Owning sheet.
`Connected1` / `Connected2`	enum	always	Endpoint kind — see enum below.
`Bus1` / `Bus2`	Int	always	Connected bus `Id` when the endpoint is a bus, else `-1`.
`Object1` / `Object2`	Int	always	Endpoint target: Pin→part `Id`; Wire→another wire `Id` in the net; Bus→bus-wire `Id`.
`SubObject1` / `SubObject2`	Int	always	Pin→pin index in the part; Wire/Bus→point index in that wire.
`HiddenPower`	Bool	always	Hidden power/ground wire.
`CanUnhide`	Bool	always	Whether a hidden wire may be unhidden.
`Arrows`	enum	if >0	`0`=None, `1`=Forward, `2`=Backward.
`Group`	Int	if >−1	Group id.
`Selected`	Bool	if set	Editor flag.

Value	Name
`0`	Pin
`1`	Wire
`2`	2 (net-junction placeholder)
`3`	Bus
`4`	Free

Each `<Point>` carries `X`/`Y` (Real, Y negated) and `Dir` (Int): `-1` unset, `0` horizontal, `1` vertical — the direction is stored on the segment's second point; the first point's `Dir` is ignored.

<DifferentialPairs> / <DifferentialPair>

Pairs the positive and negative nets of a differential pair.

```
<DifferentialPairs>
  <DifferentialPair Id="0" PosNet="4" NegNet="5" NetClass="1" CustomColor="Y"
WireColor="16711680">
    <Name>USB_D</Name>
  </DifferentialPair>
</DifferentialPairs>
```

Attribute	Type	Written	Description
`Id`	Int	always	Pair index (load-bearing on merge/exchange import).
`PosNet` / `NegNet`	Int	always	Positive / negative net `Id`.
`NetClass`	Int	always	Positional net-class index.
`CustomColor`	Bool	if set	Use a custom color; when set, `WireColor` follows.
`WireColor`	Int	if custom	Custom wire color.
`<Name>`	Text	always	Pair name (child element).

> **Note.** `Enabled` is not produced for a diff pair — the export loop is gated on the enabled flag, so a disabled pair is pruned (no element). The presence of the element itself is the "enabled" signal. The reader still accepts `Enabled` on hand-authored XML.

<Buses> / <Bus> and <BusConnectors> / <BusConnector>

A bus lists its member nets and its bus wires. Bus connectors are the named ports that join a bus to nets.

```
<Buses>
  <Bus Id="0">
    <Name>DATA</Name>
    <Nets><Net NetId="4" ConnectionNumber="0"/><Net NetId="5"
ConnectionNumber="1"/></Nets>
    <Wires>
      <Wire Id="0" Sheet="1" Connected1="Bus Connector" Bus1="-1" Object1="2"
SubObject1="0"
            Connected2="Bus Wire" Bus2="-1" Object2="1" SubObject2="0">
        <Points><Point X="10" Y="-20" Dir="-1"/>…</Points>
      </Wire>
    </Wires>
  </Bus>
```

```
</Buses>
<BusConnectors>
  <BusConnector Id="0" Sheet="1" X="10" Y="-20" Connected="Y"><Name>DATA</Name></
BusConnector>
</BusConnectors>
```

<Bus>

Attribute	Type	Written	Description
`Id`	Int	always	Bus id.
`Locked`	Bool	if set	Locked.
`Enabled`	Bool	—	Not emitted by a native save (loop gated on enabled). Reader-accepted delete flag.
`<Name>`	Text	always	Bus name (child element).

`<Nets>` holds `<Net NetId="…" ConnectionNumber="…"/>` (member net `Id` and its ordinal in the bus). `<Wires>` holds bus `<Wire>`s with the same connection model as net wires (`Id`, `Sheet`, `Connected1/2`, `Bus1/2`, `Object1/2`, `SubObject1/2`, optional `Group`/`Selected`) and `<Points>`. Bus-wire endpoint kinds: `2`=Bus Wire, `4`=Bus Connector, all other values render as Free.

<BusConnector>

Attribute	Type	Written	Description
`Id`	Int	always	Connector id.
`Sheet`	Int	always	Owning sheet.
`X` / `Y`	Real	always	Position (Y negated).
`Connected`	Bool	always	Whether the connector is attached to a bus.
`Enabled`	Bool	—	Not emitted by a native save (loop gated on enabled). Reader-accepted delete flag.
`Group`	Int	if >−1	Group id.
`Selected` / `Locked`	Bool	if set	Editor flags.
`<Name>`	Text	always	Port name (child element).

<Shapes> / <Shape>

Free graphics and text. Attribute set varies by `Type`: line-style attributes for outlines/fills, picture attributes for pictures, and the full font family for text.

```
<Shape Id="0" Type="Rectangle" Sheet="0" LineWidth="0.25" Color="9079434"
Selected="N">
```

```
  <Points><Point X="86.995" Y="21.59"/><Point X="120" Y="40"/></Points>
</Shape>
<Shape Id="1" Type="Text" Sheet="0" Angle="0" HorzAlign="Left" VertAlign="Top"
TextAlign="Left"
       FontVector="Y" FontMono="N" FontSize="10" FontSizeFloat="10"
FontWidth="-2" FontScale="1"
       FontColor="0" LineSpacing="1.2" TextWidth="18" TextHeight="4">
  <Points><Point X="10" Y="-5"/></Points>
  <FontName>Tahoma</FontName>
  <TextLines><TextLine>Power</TextLine></TextLines>
</Shape>
```

Attribute	Type	Written	Description
`Id`	Int	always	Shape index.
`Type`	enum	always	See enum below.
`Sheet`	Int	always	Owning sheet.
`LineWidth`	Real	all types except Text and Picture, if >−1	Outline width — written for fills too (FillRect, FillObround).
`Color`	Int	all types except Text and Picture	Line/fill color.
`NetId`	Int	Text shapes, if ≥0	Bound net (net-name label).
`BusId`	Int	Text shapes, if ≥0	Bound bus (bus label).
`Angle`	Real	Text and Picture	Text/picture rotation.
`PictureWidth` / `PictureHeight`	Real	Picture only	Picture size.
`PictureProportions`	Bool	Picture only	Keep aspect ratio.
`PictureFlipped`	Bool	Picture only	Picture mirrored.
`PictureTransparent`	Int	Picture only	Transparent-color value.
`HorzAlign` / `VertAlign`	enum	Text and Picture	Anchor alignment (Left/Center/Right; Top/Center/Bottom).
`TextAlign`	enum	Text only	Paragraph alignment.
`FontVector`	Bool	Text only	Vector vs TrueType font; `FontMono` follows when `Y`.
`FontSize` / `FontSizeFloat`	Int / Real	Text only	Font size (float twin wins on import).

Attribute	Type	Written	Description
`FontWidth` / `FontScale` / `FontColor` / `LineSpacing`	Real/ Real/ Int/ Real	Text only	Stroke width, scale, color, line spacing.
`TextWidth` / `TextHeight`	Real	Text only, if has lines	Computed text extent (informational).
`Group`	Int	if >−1	Group id.
`Selected` / `Locked`	Bool	if set	Editor flags.

Value	Name
`0`	None
`1`	Line
`2`	Arc
`3`	Arrow
`4`	Rectangle
`5`	FillRect
`6`	Obround
`7`	FillObround
`8`	Polyline
`9`	Polygon
`10`	Text
`11`	Picture

Child elements: `<Points>`/`<Point X Y>` (geometry). A Text shape also writes `<FontName>` and a `<TextLines>`/`<TextLine>` list; a Picture shape writes a `<PictureFile>` (`<Path>`/`<Var>`).

> **Note.** `LineWidth` and `Color` are written for filled shapes (FillRect, FillObround) too — the only exclusions are Text and Picture. `Enabled` is not emitted (disabled shapes are pruned); the reader accepts it as a delete flag.

<Tables> / <Table> / <Cell>

Free / BOM tables. Table-level attributes carry the default cell font; each cell can override it.

```
<Table Id="0" Sheet="1" X1="10" Y1="-10" X2="90" Y2="-40" Orientation="0"
       CellWidth="20" CellHeight="6" TextAlign="Left" FontVector="Y" FontMono="N"
       FontSize="8" FontSizeFloat="8" FontWidth="-2" FontScale="1" FontColor="0"
```

```
LineSpacing="1.2">
  <FontName>Tahoma</FontName> <Name>BOM</Name>
  <ColWidths><Item>20</Item>…</ColWidths>
  <RowHeights><Item>6</Item>…</RowHeights>
  <Cells>
    <Cell>
      <Cell X1="10" Y1="-10" X2="30" Y2="-16" TextAlign="Left" FontVector="Y"
FontMono="N"
            FontSize="8" FontSizeFloat="8" FontWidth="-2" FontScale="1"
FontColor="0"
            LineSpacing="1.2" TextWidth="12" TextHeight="4">
        <FontName>Tahoma</FontName>
        <TextLines><TextLine>RefDes</TextLine></TextLines>
      </Cell>
    </Cell>
  </Cells>
</Table>
```

Attribute	Type	Written	Description
`Id`	Int	always	Table index.
`Sheet`	Int	always	Owning sheet.
`HideBorder`	Bool	if set	Suppress the outer border.
`X1`/`Y1`/`X2`/`Y2`	Real	always	Table rectangle (Y negated).
`Orientation`	enum	always	`0`/`1`/`2`/`3` = 0/90/180/270.
`CellWidth` / `CellHeight`	Real	always	Default cell size.
`TextAlign`	enum	always	Default cell alignment (Left/Center/Right).
`FontVector` / `FontMono`	Bool	always / vector only	Default cell font kind; `FontMono` only when vector.
`FontSize` / `FontSizeFloat`	Int / Real	always	Default cell font size.
`FontWidth` / `FontScale` / `FontColor` / `LineSpacing`	Real/ Real/ Int/ Real	always	Default cell font metrics.
`Group`	Int	if >−1	Group id.
`Selected` / `Locked`	Bool	if set	Editor flags.
`<FontName>` / `<Name>`	Text	always	Default font family; table name.

Cells are written as a `<Cells>` list of rows, each row a `<Cell>` containing the row's `<Cell>` entries. A cell carries its own rectangle (`X1`/`Y1`/`X2`/`Y2`), `TextAlign`, the full font family (`FontVector`, `FontMono` when vector, `FontSize`/`FontSizeFloat`/`FontWidth`/`FontScale`/`FontColor`/`LineSpacing`, `<FontName>`), and, when it has text, `TextWidth`/`TextHeight` plus a `<TextLines>`/`<TextLine>` list.

> **Note.** The row and cell wrappers share the tag name `<Cell>` — the outer level is a row, the inner level a cell. `Enabled` is not emitted (disabled tables are pruned); the reader accepts it as a delete flag.

<Simulator> — SPICE setup

Written only when the project has simulation data. Carries the analysis type, plotted signals, per-analysis parameter blocks, a digital-simulation block, and four SPICE solver-options blocks.

```
<Simulator Type="Transient" GndNetId="0">
  <Signals>
    <Signal Type="Voltage" Id="4" Plotted="Y" Visible="Y" Color="255">
      <DisplayName>V(out)</DisplayName>
    </Signal>
  </Signals>
  <Transient><StartTime>0</StartTime><Step>1u</Step><StopTime>1m</StopTime></
Transient>
  <Digital><StartTime>0</StartTime><Step>1n</Step><StopTime>1u</
StopTime><GroupBuses>Y</GroupBuses></Digital>
  <OptionsGeneral NoPrintStat="N" NoPrintModelParams="N"><Temp>27</
Temp><TempNomMeasure>27</TempNomMeasure></OptionsGeneral>
  <OptionsDC ConductanceSteps="0" SkipConductanceStepping="N" Iteration="100"
TransferCurveIteration="50" StartFromConductanceStepping="N">…</OptionsDC>
  <OptionsAC NoOPAnalysis="N"/>
  <OptionsTransient LowIteration="0" PointIteration="10" TotalIteration="5000"
SourceStepping="0" MethodMaxOrder="2" Method="1">…</OptionsTransient>
  <GndNetName>GND</GndNetName>
</Simulator>
```

Attribute	Type	Written	Description
`Type`	enum	always	Analysis type — see enum below.
`GndNetId`	Int	always	Ground net `Id` (`-1` for none).

Value	Name
`1`	Small-Signal AC
`2`	DC Transfer Function
`4`	Noise
`9`	Transient

Value	Name
10	Digital
11	Options

Child blocks (each written when it has data): <Signals> of <Signal Type Id Plotted Visible Color> with a <DisplayName> (Type: 0=Current, 1=Voltage); <SmallSignalAC>, <DCTransferFunc>, <Transient>, <Noise> parameter blocks; and <Digital> (<StartTime>/<Step>/<StopTime>/<GroupBuses>).

SPICE solver-options blocks

Four blocks written unconditionally within <Simulator>, carrying the numeric SPICE options as text/attribute values.

- <OptionsGeneral>: NoPrintStat, NoPrintModelParams (Bool attrs); <Temp>, <TempNomMeasure> (text).
- <OptionsDC>: attrs ConductanceSteps, SkipConductanceStepping, Iteration, TransferCurveIteration, StartFromConductanceStepping; text children <AbsCurrentTolerance>, <MinConductance>, <PivotRation>, <PivotTolerance>, <RelativeError>, <GroundResistor>, <VoltageError>.
- <OptionsAC>: NoOPAnalysis (Bool attr).
- <OptionsTransient>: attrs LowIteration, PointIteration, TotalIteration, SourceStepping, MethodMaxOrder, Method (0=unknown, 1=default/Trapezoidal, 2=Gear); text children <ChargeTolerance>, <SourceSteppingChange>, <ErrorTolerance>, <TrapezoidalDampingFactor>.

The block closes with a <GndNetName> text element.

<Groups> / <Group>

Registry of object groups.

```
<Groups>
  <Group Id="0" Selected="N"/>
  <Group Id="1" Enabled="N"/>
</Groups>
```

Attribute	Type	Written	Description
Id	Int	always	Group id; objects reference it via Group.
Enabled	Bool	N only, for a deleted group	Written as N when the group is removed; otherwise omitted.
Selected	Bool	if set	Editor flag (only Y written).

> **Note. Enabled across the dialect.** Enabled="N" is the import/exchange **delete flag**, honored by the plug-in importer for nets, shapes, tables, buses, bus connectors, differential pairs,

groups, *and* parts. On a native save it is genuinely emitted only for a deleted `<Net>` and a deleted `<Group>`; for shapes, tables, buses, bus connectors, diff pairs, and parts the disabled object is simply pruned (no element), so a written `Enabled="N"` comes only from hand-authored or plug-in XML. **Wires carry no `Enabled` at all** — remove a wire by dropping it from its net's `<Wires>` list (the list is subtree-replaced when present), or remove the whole net with its wires via `Enabled="N"` on the `<Net>`.