# Differences Between MaterialX Specification v1.35 and v1.36

Updated August 6, 2018

## New Standard Operators

The following new standard operator nodes have been added:

- Trig functions: **sin**, **cos**, **tan**, **asin**, **acos**, **atan2**
- Other math functions: **sign**, **ceil**, **sqrt**, **ln**, **exp**
- Matrix functions: **invert**, **transpose**, **determinant**
- Vector transform functions: **rotate**, **transformpoint**, **transformvector**, **transformnormal**
- Array manipulation functions: **arrayappend**
- Adjustment node: **curveadjust**
- Color conversion functions: **rgbtohsv**, **hsvtorgb**
- Type conversion function: **convert**

Additionally:

- Basic math operations have been expanded to support matrix33 and matrix44 types.
- The <multiply> and <divide> operators now support 3x3 and 4x4 matrix*matrix.
- The various Blend and Merge operators now have an additional float "mix" input, allowing mixing the result back with the "bg" input.
- The "exponent" and "pack" operators have been renamed to "power" and "combine" respectively, to better match the naming of these functions in common systems.

## Simplified Mechanism for Inheritance

Inherited materials and looks now use `inherit` *attributes* in the <material> or <look> element rather than separate child <materialinherit> or <lookinherit> elements. There is also now an `inherit` attribute for <nodedef>, allowing nodedefs to inherit baseline definitions from other nodedefs: this is handy for declaring target renderer-specific versions of nodes with target-specific (custom) parameters inheriting from a canonical definition of the node.

## New Syntax for Collection Creation

The syntax for creating collections has been changed: instead of an arbitrary set of child <collectionadd> and <collectionremove> elements, the <collection> element itself now has `includegeom`, `includecollection` and `excludegeom` attributes. This was done both to enforce "all adds before any removes" ordering, and to be compatible with the collection-definition semantics of USD.

## Tokens and Image Filename Substitutions

A new mechanism for defining and passing string values for use in Image Filename Substitutions has been defined: <geominfo>s and nodes/<nodedef>s can now define **tokens** using a new <token> element, leaving <geomattr>s to define uniform or declare varying values on geometries only accessible in nodegraphs via <geomattrvalue> nodes. This simplifies the data model for

these important special-case objects, and limits the need for MaterialX to be aware of or have access to the actual geometry state. We differentiate between tokens defined per-geometry in a <geominfo> ("geometry tokens") and those defined using a new <bindtoken> element within a <shaderref> and passed like a parameter to a node ("interface tokens").

The syntax for Image Filename Substitutions has also been changed for easier and more robust parsing. The new syntax is <*geomtokenname*>, [*interfacetokenname*] and {*hostattribute*}, replacing %*geomattr* and $*hostattribute*. This change also means that the syntax for referencing a Mari-style Udim is now <UDIM> instead of the previous %UDIM (conveniently matching the native syntax of many renderers), and $frame is now {frame}.

**Node Versioning Support**

It is now possible to define or request a specific version for a node, thus allowing multiple versions to live within a document and to reference specific versions in node instantiations. The new `version` attribute can be associated with:

- A node invocation or shaderref (to request a specific version of the node)
- A nodedef (to say this is the definition of version "9" of the node, and optionally declare this nodedef to be the default version)

The "version" attribute works pretty much exactly the same way that the "target" attribute does.

**Variants Instead of MaterialVars**

The MaterialVar mechanism has been removed, and a new Variant mechanism has been defined to replace it. Variants are named lists of parameter values grouped together into VariantSets, which can then be applied in material assignments using <variantassign> elements to set values for those parameters in the material from within a Look. The new mechanism is hopefully both simpler, more useful, and more similar to material variation mechanisms used in various applications.

**New Attributedef Element**

A new <attributedef> element has been added, which can be used to formally declare the name, type, default value, and optional target for custom attributes. One can also optionally declare a list of element types that the custom attribute may be used in.

**Publicname and Overrides Removed**

Previous versions of the MaterialX Specification allowed the definition of `publicname` attributes for parameters and inputs of nodes within a nodegraph, which could then be externally assigned a value in a material using an <override> element. As of version 1.36, this functionality has been removed in favor of instead using the existing <nodedef> parameter interface for nodegraphs.

**"Require" Declarations Removed**

Previous versions of the MaterialX Specification outlined an "Implementation Compatibility Checking" mechanism using "require" attributes on top-level <materialx> elements to indicate broad categories of functionality that a document made use of. As of v1.36, applications are expected to do this compatibility checking themselves, as applications are likely to have far more fine-grained and nuanced understanding of their specific capabilities than the previous mechanism could afford.

**Other Changes**

- The syntax for declaring string and stringarray values has changed slightly to be more compliant with XML standards, using character entities rather than backslashes for "special characters".
- The `vdirection` global attribute has been removed, and we have clarified that MaterialX defines increasing V values as "up". Any application which treats increasing "V" values as "down" should translate any "top" or "bottom" parameters accordingly upon import and export.
- Added further clarification on the use of different colorspaces in documents, and how MaterialX expects applications to convert and pre-convert images and colors to the working colorspace.
- Removed the "channels" attribute for node elements; type conversion between nodes must be done explicitly using `convert` or `swizzle` nodes.
- Nodes and <output> elements are no longer required to be defined inside a nodegraph, and it is now allowable to bind an input in a <shaderref> directly to the named <output> element (but not to the output of any arbitrary node). The <nodegraph> element is now only required when defining a graph of nodes to be wrapped into a nodedef interface, although wrapping arbitrary graphs of nodes within a <nodegraph> is still supported.
- Multiple XIncluded <materialx> elements are explicitly permitted to specify the same namespace, so that the elements defined for a particular namespace library may be spread across multiple files.
- Added a new <viewdirection> application node, which returns the current camera view vector.
- The various image nodes now have a `filtertype` parameter, allowing the declaration of what type of filtering to use when reading textures.
- The image-space-based <scale> and <rotate2d> nodes have been removed, and a new general 2D or 3D-about-an-axis <rotate> node has been added.
- Descriptions of certain standard nodes (<triplanarprojection>, <contrast> and <saturate>) which have straightforward nodegraph-based implementations have been moved to the new **MaterialX: Supplemental Notes** document.
- The standalone <hueshift> node has been removed, replaced with a new <hsvadjust> supplemental node.

- The <remap> node's original functionality is now provided by the new <range> supplemental node, while <remap> becomes a straight linear remapping from one range to another.
- The various Blend and Merge operators now support a "mix" input, allowing the result to be mixed back with the "bg" input.
- The `color3` variants of <premult> and <unpremult> have been removed, as they were exactly the same as existing variants of the <multiply> and <divide> nodes.
- Added a new standard attribute `doc` which can be used with *any* element type, e.g. for documentation of nodedefs. The new `doc` attribute is also now used for node parameters/inputs, so the previous `helptext` attribute has been removed.
- Enum parameters for types other than string are now permitted, using the existing (but now renamed) `enum` attribute for the labels along with a new `enumvalues` attribute for the values. Enum and enumvalues are defined in nodedef <parameter>s, although <parameter>s within implementations are now allowed to change the `enumvalues` mapping for specific targets.
- The `nodecategory` attribute for nodedefs is now called `nodegroup`, so as to not be easily confused with the Node::getCategory() method in the library API, which is used to return a node's "node" category (e.g. "add", "noise3d" or "mix").
- Nodedefs can now specify an `internalgeomprops` stringarray attribute to define a list of geometric properties that an externally-defined node shader expects to be able to access.
- Nodedef parameters, inputs and tokens can now define `uifolder` and `uiname` attributes to specify a hierarchical folder structure for the node interface.
- `Uimin` and `uimax` attributes are now defined within parameters and inputs of nodedefs, not typically in node invocations.
- Implementation elements and parameters/inputs within them may now specify a separate `implname`, an alternative target-specific name for the node, shader, parameter or input.
- We now explicitly recommend adding custom target-specific parameters and inputs to nodes by defining a target-specific nodedef which inherits from the master definition and then adds target-specific parameters or inputs, rather than adding these parameters to invocations of nodes with a `target` attribute.
- Geometry specifications for <geominfo> and building collections are now explicitly allowed to contain non-leaf "scene location" paths, and we have clarified that assignments to non-leaf scene locations will effectively apply to all child geometries.
- Other minor edits, document reorganizations and clarifications.