



# Validation Framework for OpenUSD

Aaron Luk | ASWF USD Games WG | 23 Aug 2023

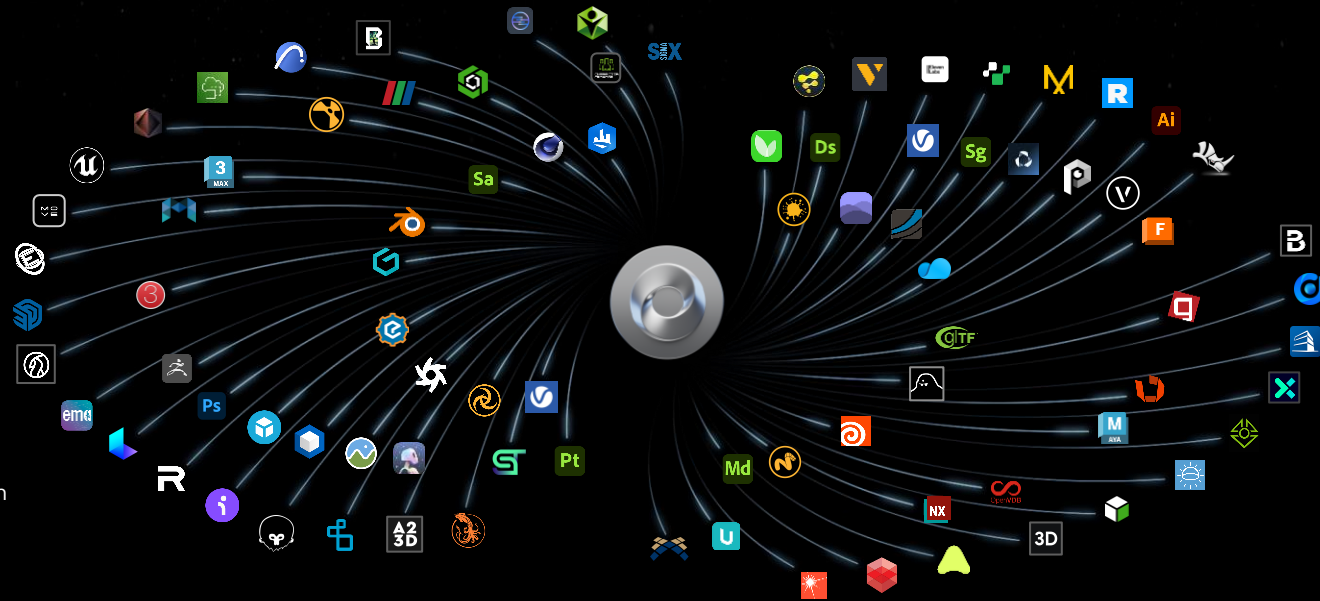
# NVIDIA OMNIVERSE

OpenUSD-Native Platform for Describing, Simulating and Collaborating Across Tools

Connects World's Largest Tool Ecosystems

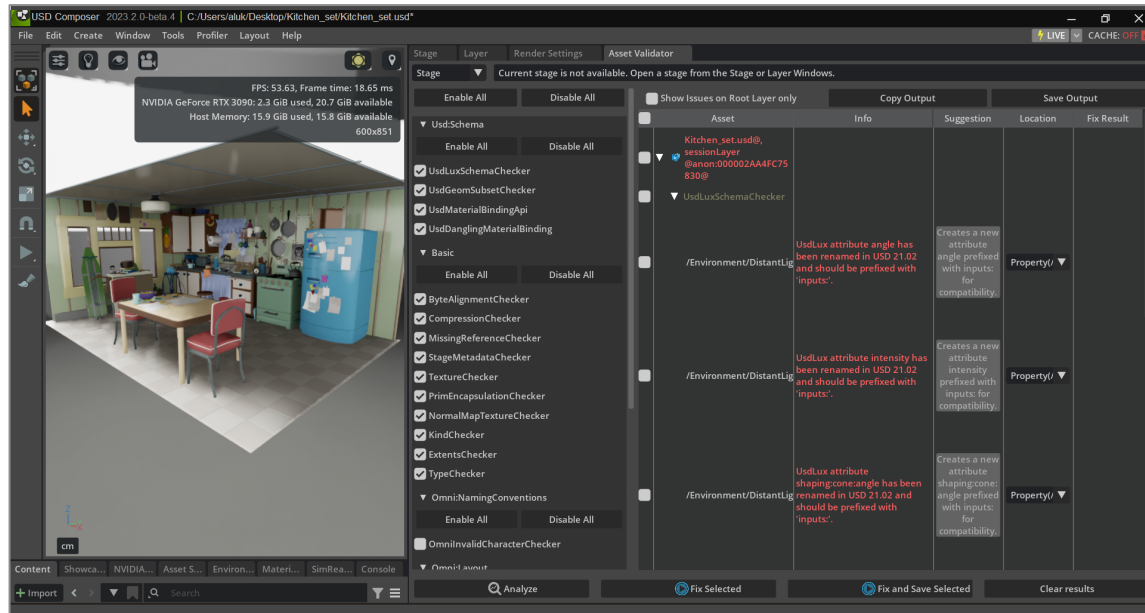
Built-In Physics and Generative AI

Platform for End-to-End Industrial Digitalization

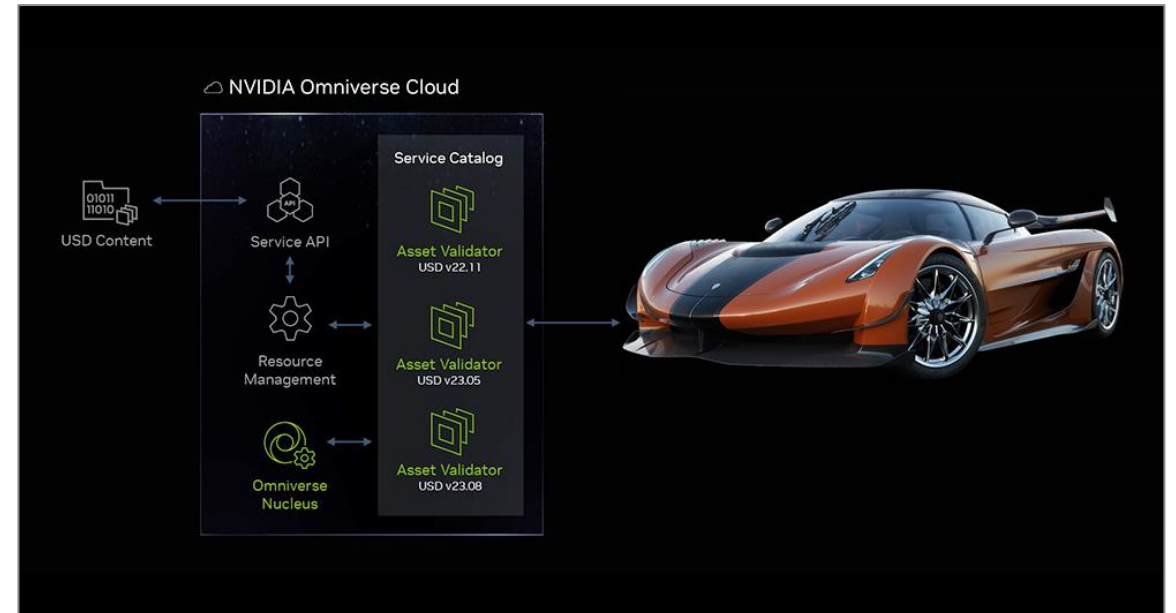


# OpenUSD Validation Framework

[https://docs.omniverse.nvidia.com/extensions/latest/ext\\_asset-validator.html](https://docs.omniverse.nvidia.com/extensions/latest/ext_asset-validator.html)



Local Workstation or VM



RunUSD – Omniverse Cloud API

# OpenUSD Validation Framework

[https://docs.omniverse.nvidia.com/extensions/latest/ext\\_asset-validator.html](https://docs.omniverse.nvidia.com/extensions/latest/ext_asset-validator.html)

- **Compliance** – *usdchecker, usdfixbrokenpxrschemas*
  - Enforce and mitigate content incompatibilities as USD data models rapidly evolve
  - e.g., UsdLux properties now require “inputs:” prefixes to conform with UsdShadeConnectableAPI
  - e.g., UsdShade material bindings now require UsdShadeMaterialBindingAPI to be applied
- **Recommendations** – custom rules
  - Potentially site-specific
  - e.g., single concrete root prim as layer default for simplicity

Note that validation often requires context that is not available until the entire stage is composed!

# Validation in Context

## Enforcing Data Specifications and Recommendations

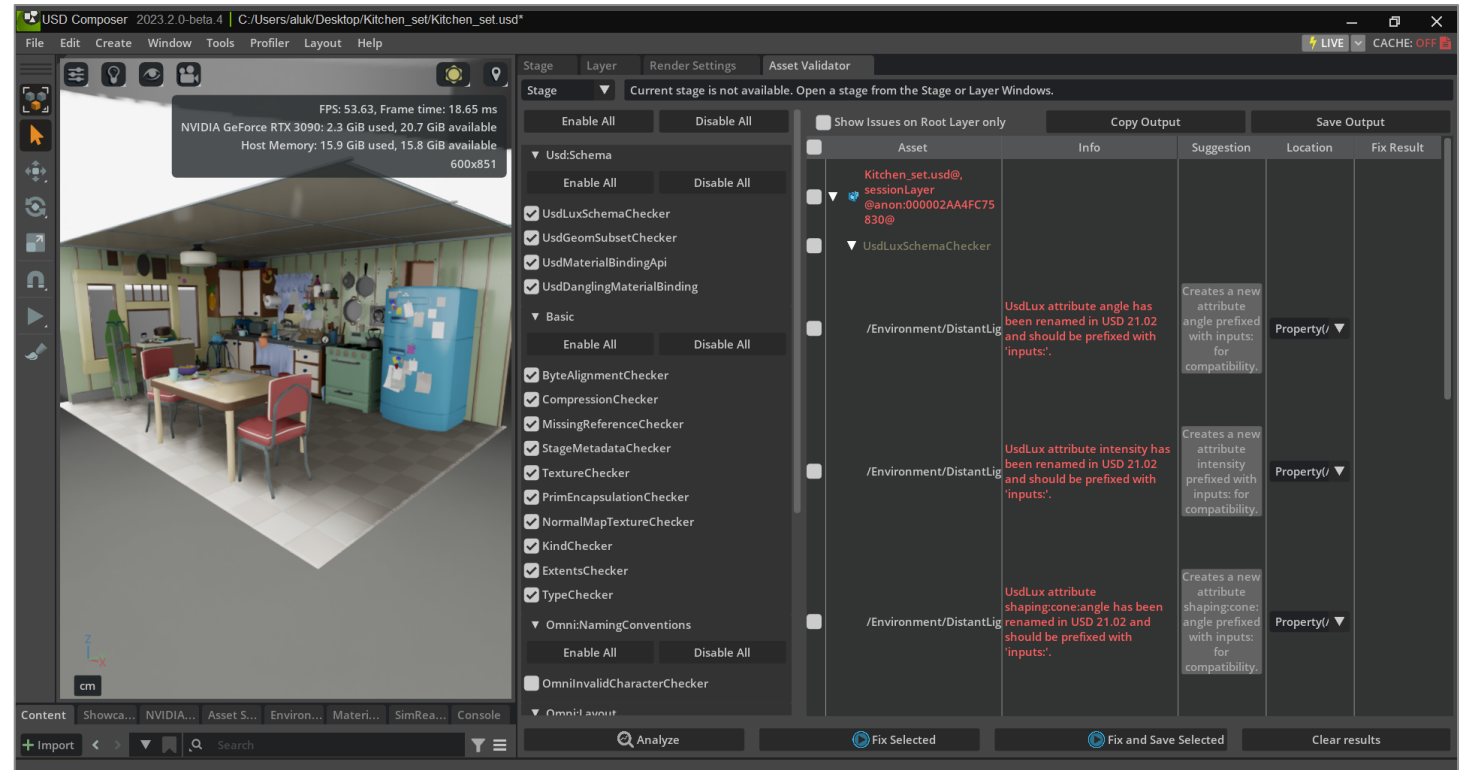
- **e.g., Nested gprim s are “illegal”**
  - Not enforced at authoring time
  - Sdf and Usd API are not gprim-aware
  - Key imaging behaviors such as visibility “assume” that gprim s are leaves in the composed stage hierarchy
  - Runtime behavior is not defined for nested gprim s
- **e.g., Recommended Model Hierarchy via kinds**
  - kind=component as leaves
  - kind=group can only have groups and components as children
  - Ill-formed model hierarchies may prematurely prune at runtime

Validation communicates standard data specifications and recommended “best practices” in the specific context of any given dataset.

# Omniverse Validation Framework

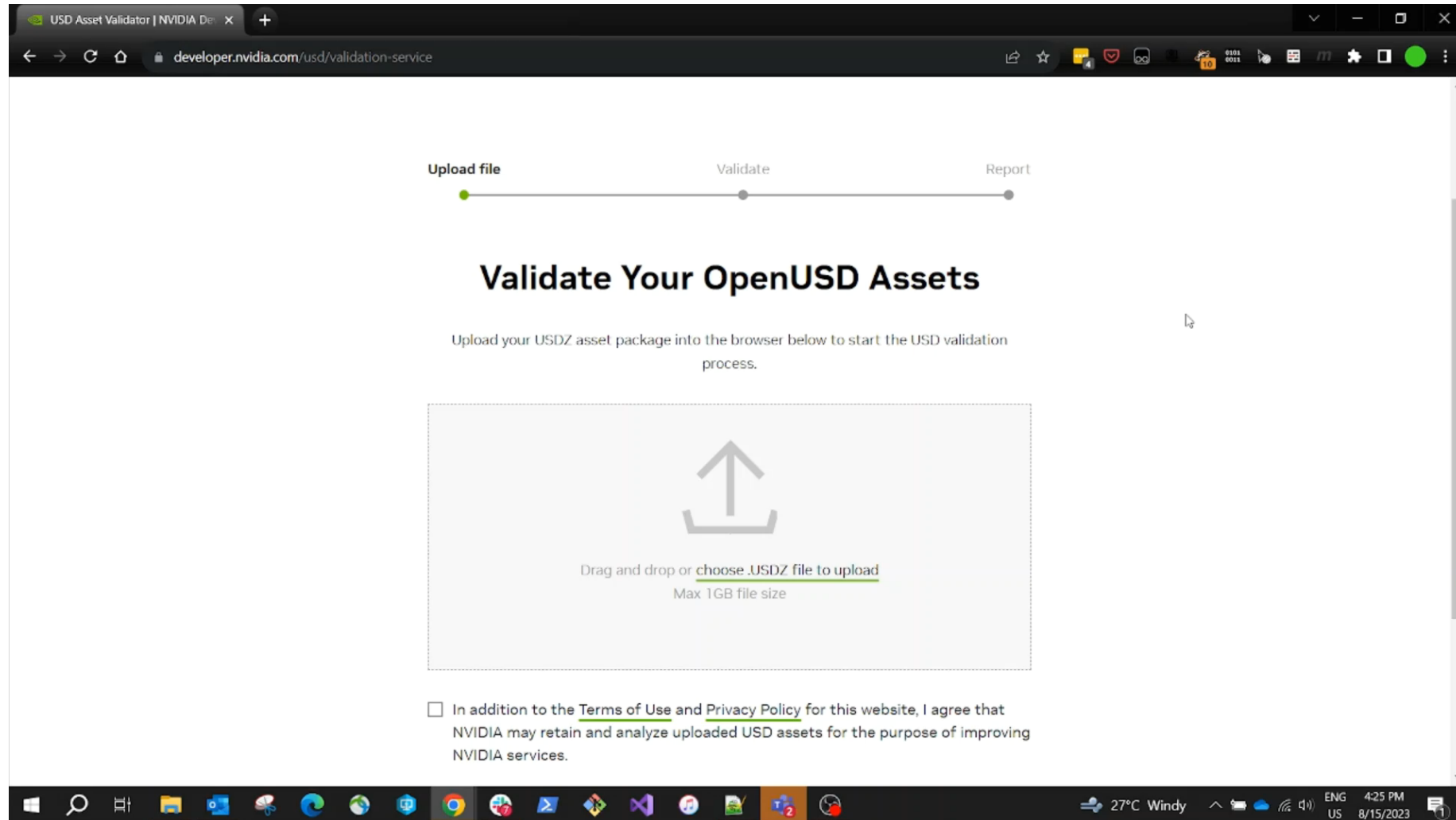
## Deployments in Action

- **In-app**
  - Interactive, cf. spell-check / linting
- **In Connectors**
  - Ensure valid exports
- **In Microservices**
  - Embed or deploy into any OpenUSD experience



# RunUSD

<https://developer.nvidia.com/usd/validator>



The screenshot shows a web browser window with the URL `developer.nvidia.com/usd/validation-service`. At the top, a progress bar indicates the workflow: **Upload file** (active), **Validate**, and **Report**. The main heading is **Validate Your OpenUSD Assets**. Below it, a text block says: "Upload your USDZ asset package into the browser below to start the USD validation process." A large dashed box contains an upload icon and the text: "Drag and drop or choose .USDZ file to upload Max 1GB file size". At the bottom, there is a checkbox with the text: "In addition to the Terms of Use and Privacy Policy for this website, I agree that NVIDIA may retain and analyze uploaded USD assets for the purpose of improving NVIDIA services." The Windows taskbar at the bottom shows the date as 8/15/2023 and the time as 4:25 PM.

# Validation Approaches

## General Content Analysis and Mitigation

- **“Offline” / asynchronous**
  - Not enforced at authoring time
  - The general approach for OpenUSD today
- **“Debug mode”**
  - Post-authoring hooks enforce validation rules
  - Pay performance penalty to discover issues ASAP
- **Hybrids, chained with other services for content conversion and optimization...**

There are many approaches to content validation.

The key thing is to communally build up a library of “business logic” to embody rules and best practices to complement forthcoming data specifications from AOUSD.



