DEVELOPER SUMMIT

March 10–13

Scene Services and the Open Indexed 3D Scene Format

Thorsten Reitz (Esri R&D Center Zurich)
Chris Andrews (Esri Inc.)

Agenda

An in-depth view at the Scene Services REST API and the Indexed 3D Scene Format

1. Web 3D as part of the ArcGIS Platform
2. Indexed 3D Scenes (i3s) & Scene Services REST API
3. i3s Persistence options
4. The i3s developer's toolkit
5. Partner implementations

Web 3D

A new Capability across the ArcGIS platform

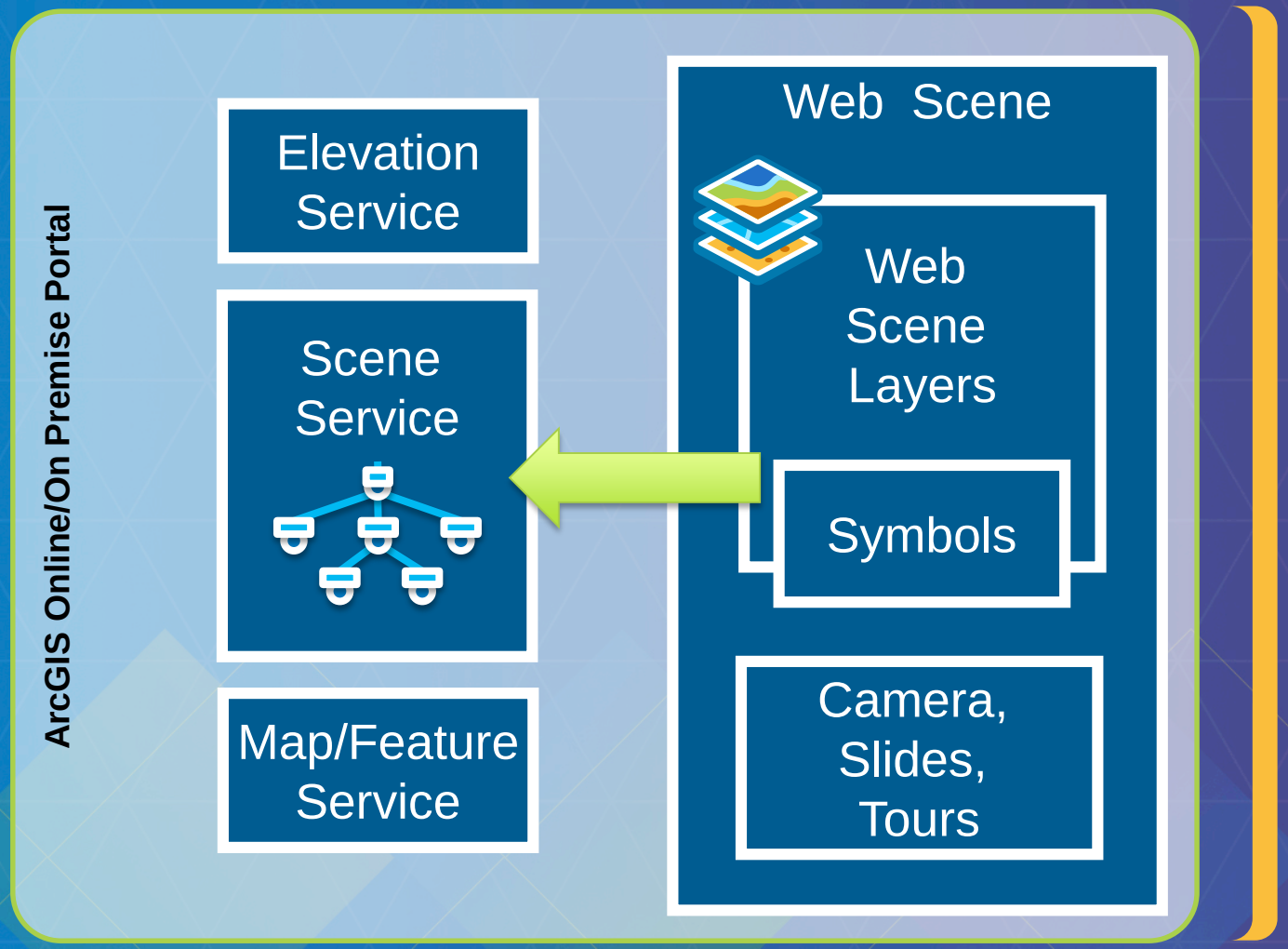
A New Way of Working in 3D

Desktop Web Device

3D across the platform

Web Scenes

The 3D Component of the ArcGIS Platform



End Users

- Pro
- Web Scene Viewer
- Explorer

Developers

- Runtime SDK
- JavaScript API

How to create Scene Services?

Geodatabase

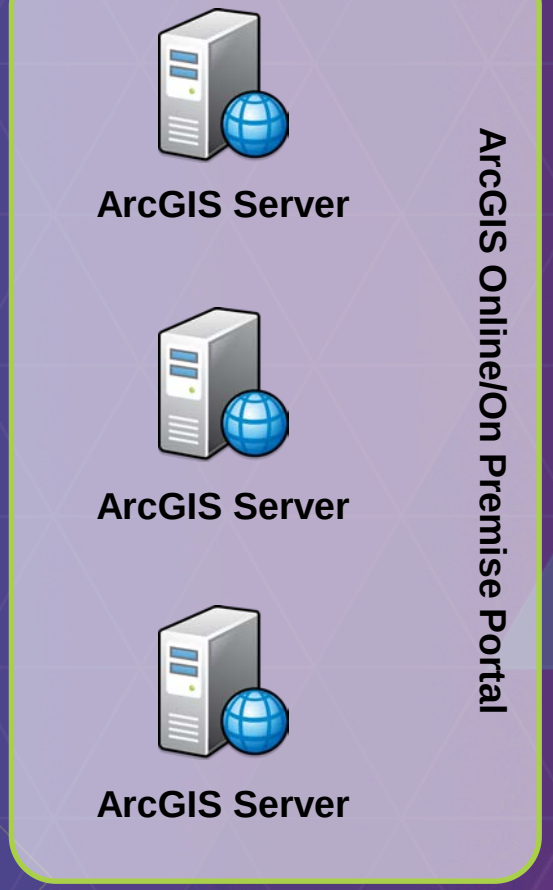
ArcGIS PRO

*.obj, CityGML

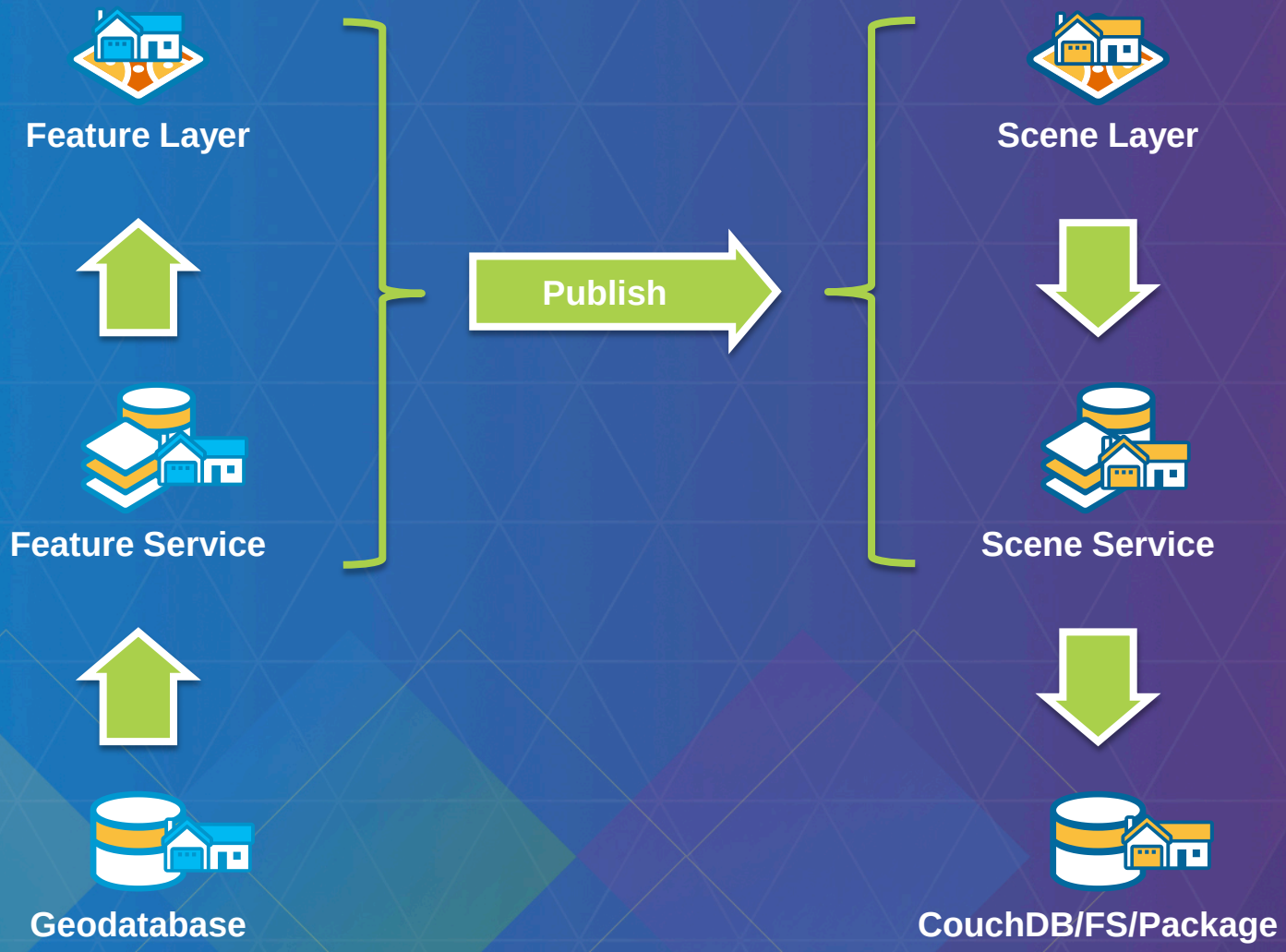
ArcGIS Online/Portal

Other Formats

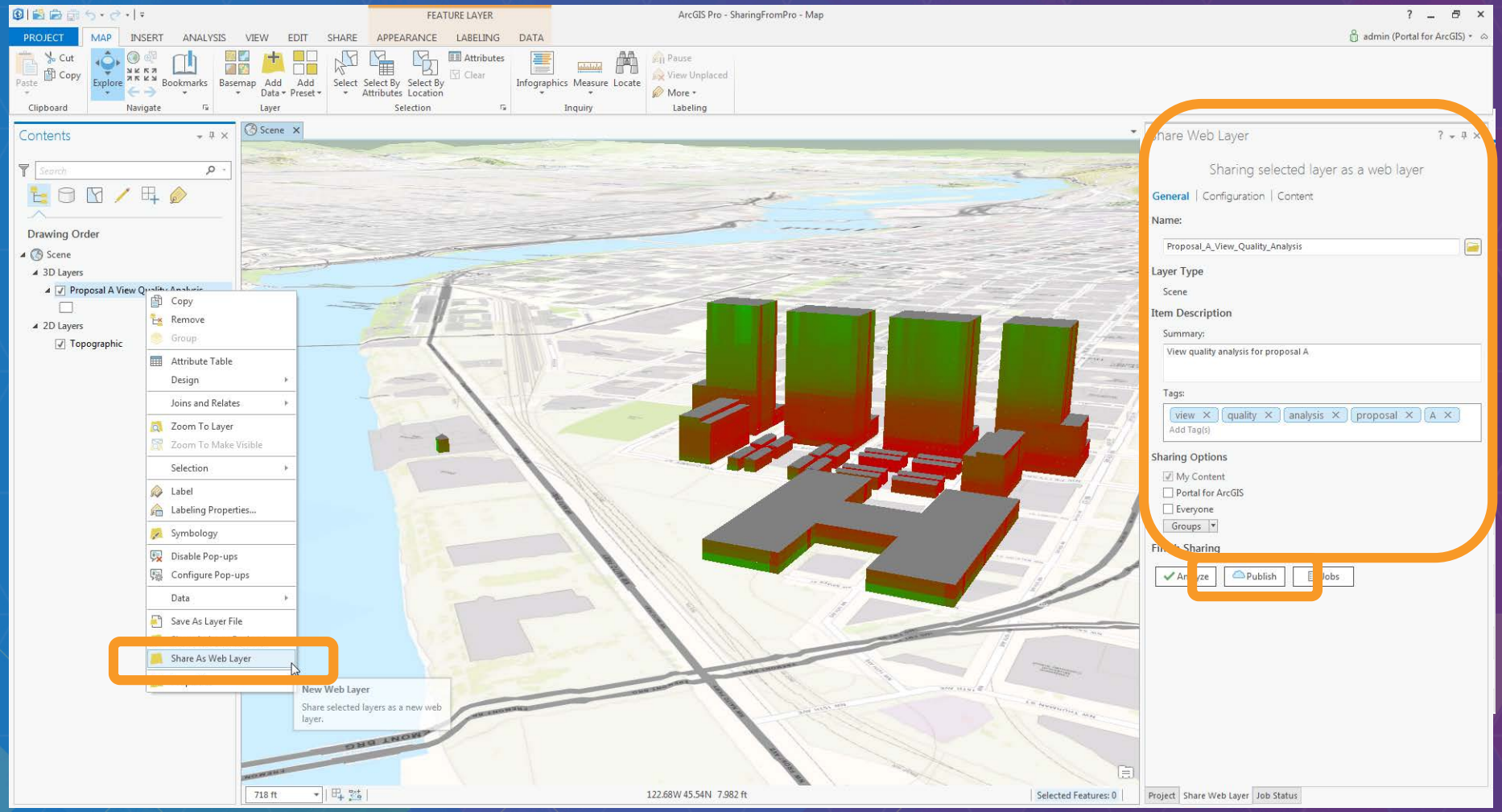
3rd Party Software



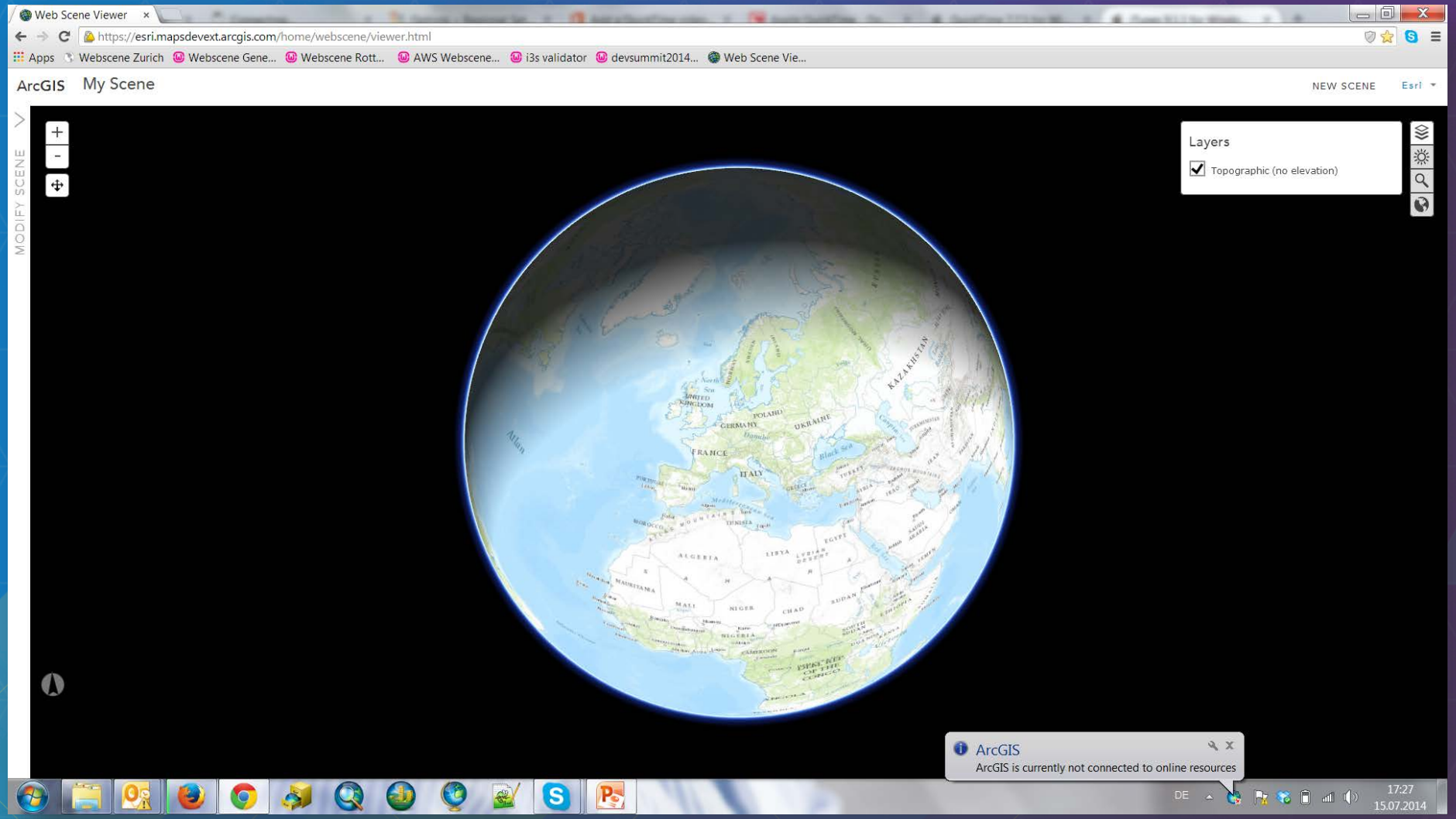
Detailed Publishing Workflow: ArcGIS Pro + Server



Publishing from ArcGIS Pro



Scene Services in Web Scenes



Demonstration

Publishing & Using Scene Services

Indexed 3D Scenes

A new open format for
multiscale geospatial 3D data

Requirements for a 3D GI visualization format

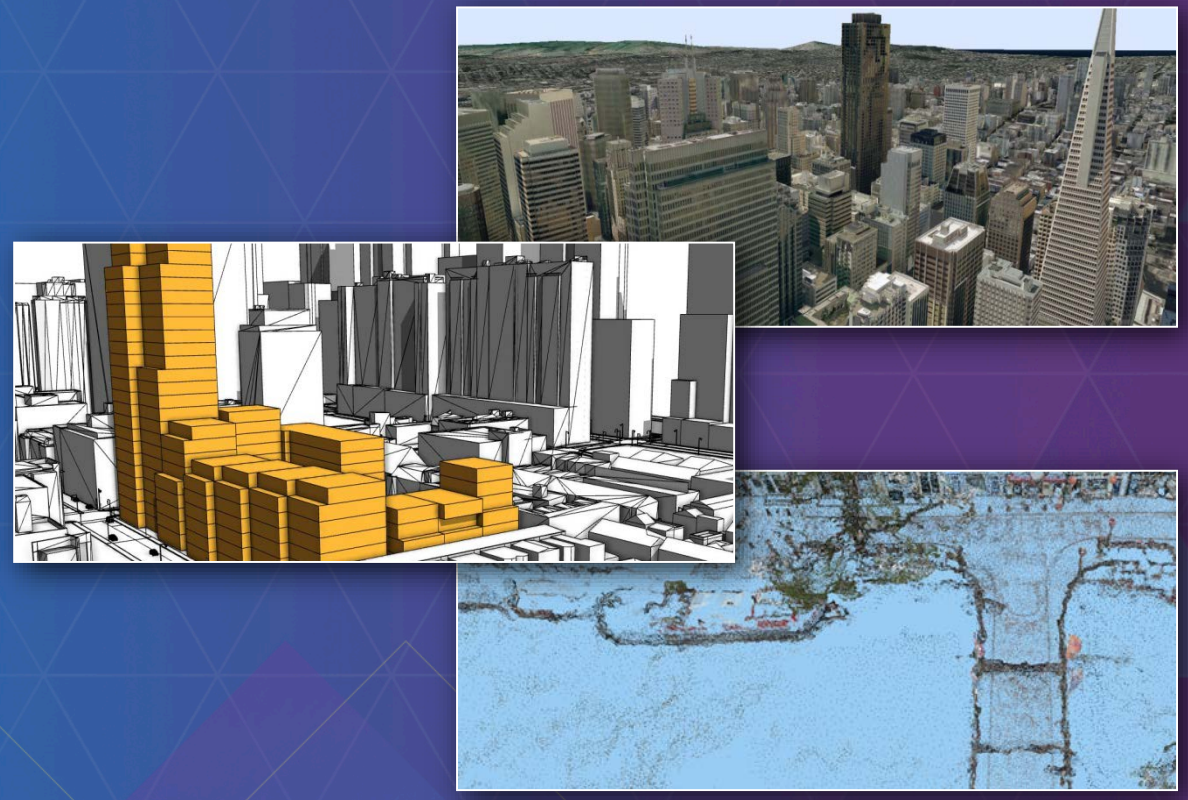
1. **Web friendly:** JSON + Typed Arrays
2. **Mobile friendly:** Works good with varying bandwidth
3. **Extensible:** Support different types of content
4. **Declarative:** Reduce required implicit knowledge
5. **Efficient:** Use spatial indexing for quick delivery
6. **Scalable:** Provide Level of Detail Support
7. **Protected:** Ensure that content is protected from theft

8. **Open:** Full Specification publicly accessible

Content Profiles

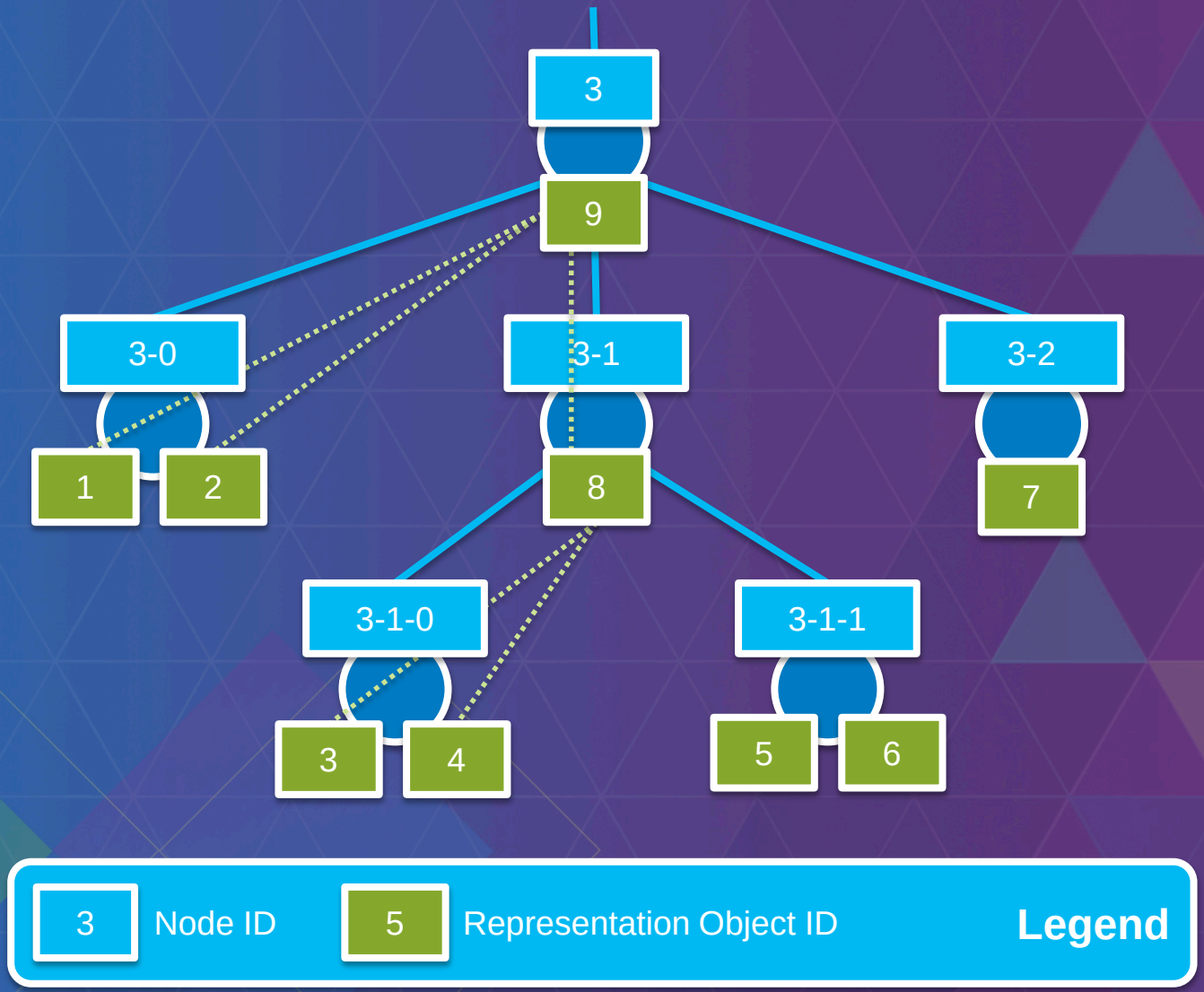
Support different geometry types through subsetting i3s

- **Individual Features**
  - **Points**
  - **Lines**
  - **Polygons**
  - **Multipatches**
- **Integrated Meshes**
- **Analytics**
- **Pointclouds**
- **Symbols**



The Key to Scalability: Indexing

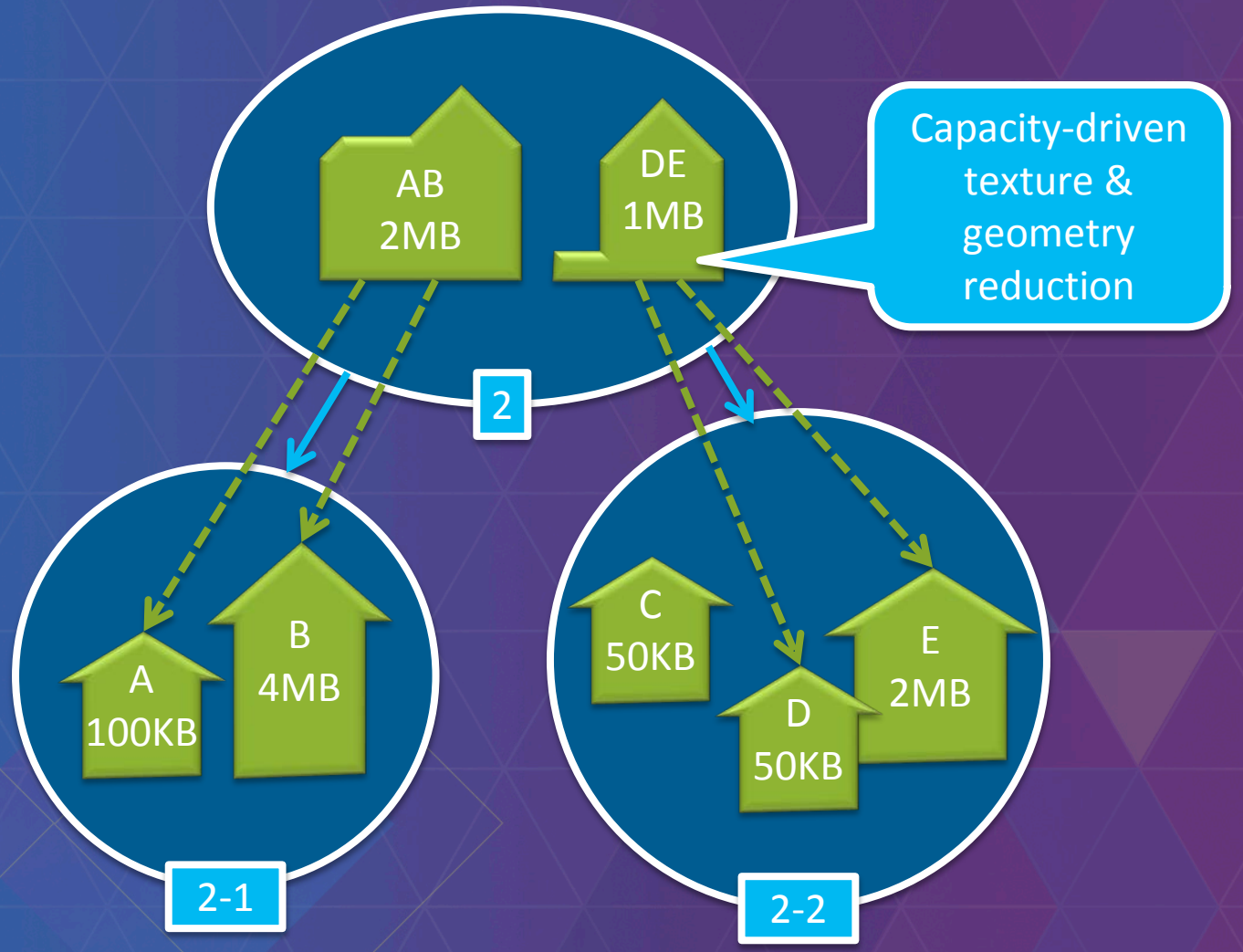
- **Adapt Index type to data**
  - **R-Tree**
  - **Quadtree, Octtree**
  - **Standard Tiling Schemes**
- **Load-Balanced**
  - **Near-Constant Data Volume per Node**



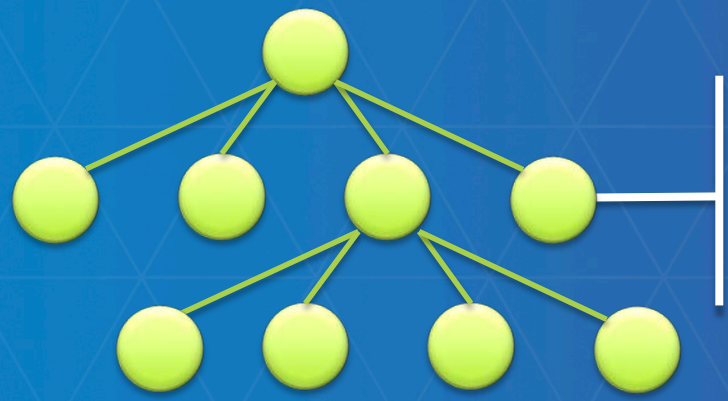
Level of Detail

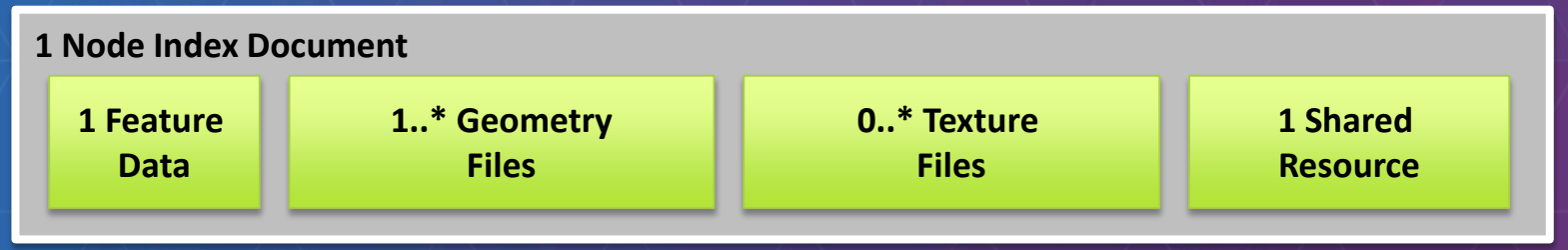
- **Feature Switching LoD**
  - ***Sparse Tree*** **with Feature-based Replacement**
  - **Use for:**
    - **Authored LoD (CityGML)**

- **Node Switching LoD**
  - ***Full Representation*** **Pyramid with Node-based Replacement**
  - **Use for:**
    - **Homogeneous contextual data (Integrated Meshes)**



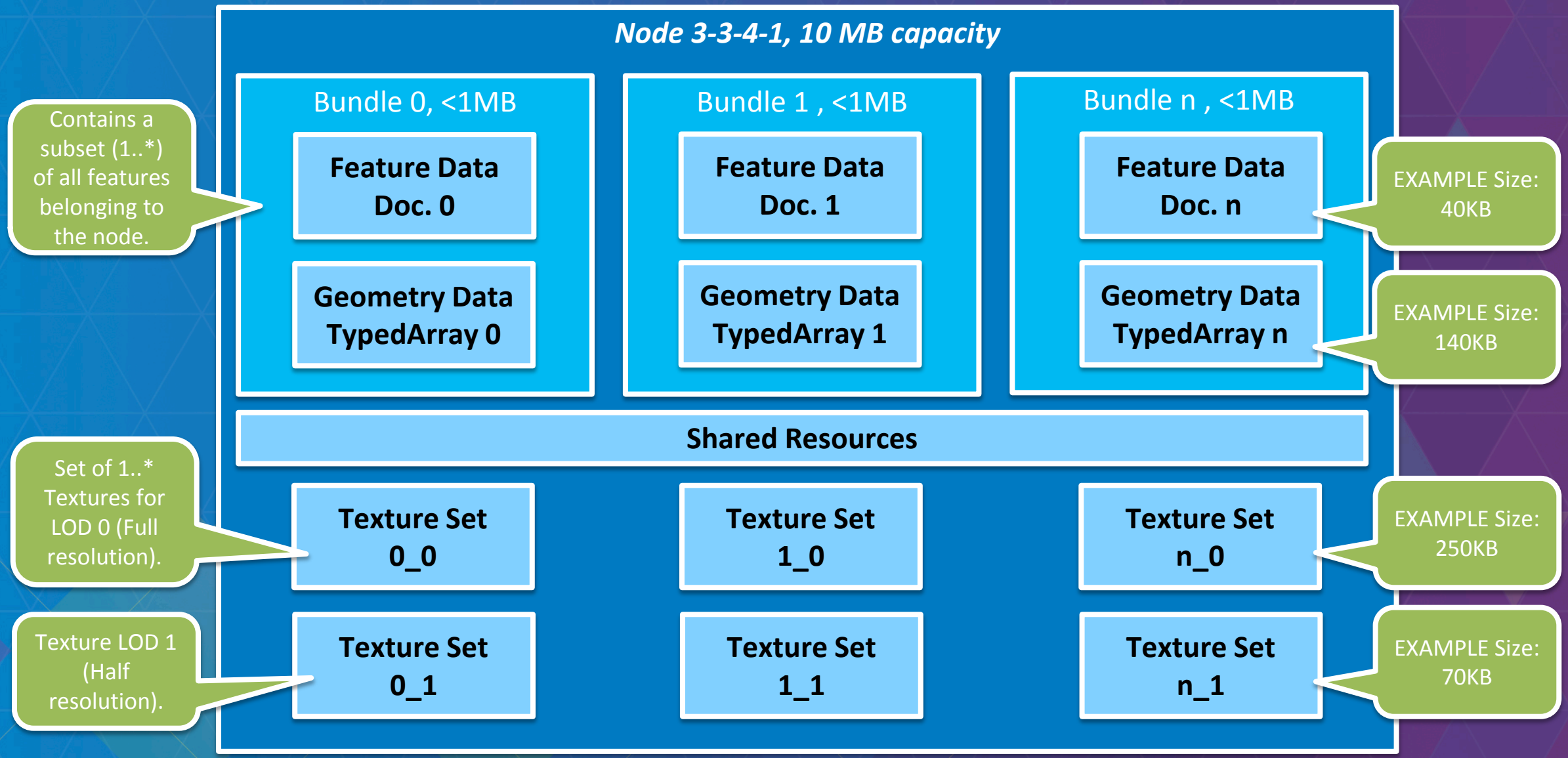
Scene Services Resources & REST API





• **Service**	/SceneServer	→ SceneServiceInfo.json
- **Layer**	/SceneServer/layers	→ 3dSceneLayer.json
- **Symbols**	/SceneServer/layers/0/symbols/0	→ FeatureData.json
- **Node**	…layers/0/nodes/0	→ NodeIndexDocument.json
- **Features**	…layers/0/nodes/0/features/0	→ FeatureData.json
- **Geometry**	…layers/0/nodes/0/geometry/0	→ Geometry Typed Array
- **Texture**	…layers/0/nodes/0/textures/1_2	→ Image with Texture LoD
- **Shared Resource**	…layers/0/nodes/0/shared	→ sharedResource.json

„Streaming“-friendly resource structure



Declarative: Geometry buffer metadata

```
"defaultGeometrySchema": { // geometry resource layout for nodes that declare the use of defaultGeometrySchema in the node index.
    "header": [ // header fields that precede the vertex data
        {
            "property": "vertexCount", // vertex count
            "type": "UInt32"
        }
    ],
    "topology": "PerAttributeArray", // one of ["PerAttributeArray", "InterleavedArray", "Indexed"]. When "Indexed", the indices must also be declared in the geo
    "vertexAttributeOrder": ["position", "normal", "uv0", "region"], // provides the order of the keys in vertexAttributes.
    "vertexAttributes": { // the vertex attributes must appear in the order that they are declared here.
        "position": { // the name of the vertex attribute; here: vertex positions
            "valueType": "Float32", // the element type, either UInt8, UInt16, UInt32,  Int16, Int32, Int64 or *Float32*, Float64
            "valuesPerElement": 3  // number of (Float32) values need to make a valid element (here a xyz position)
        },
        "normal": { // the name of the vertex attribute; here: vertex normals
```

i3s persistence options

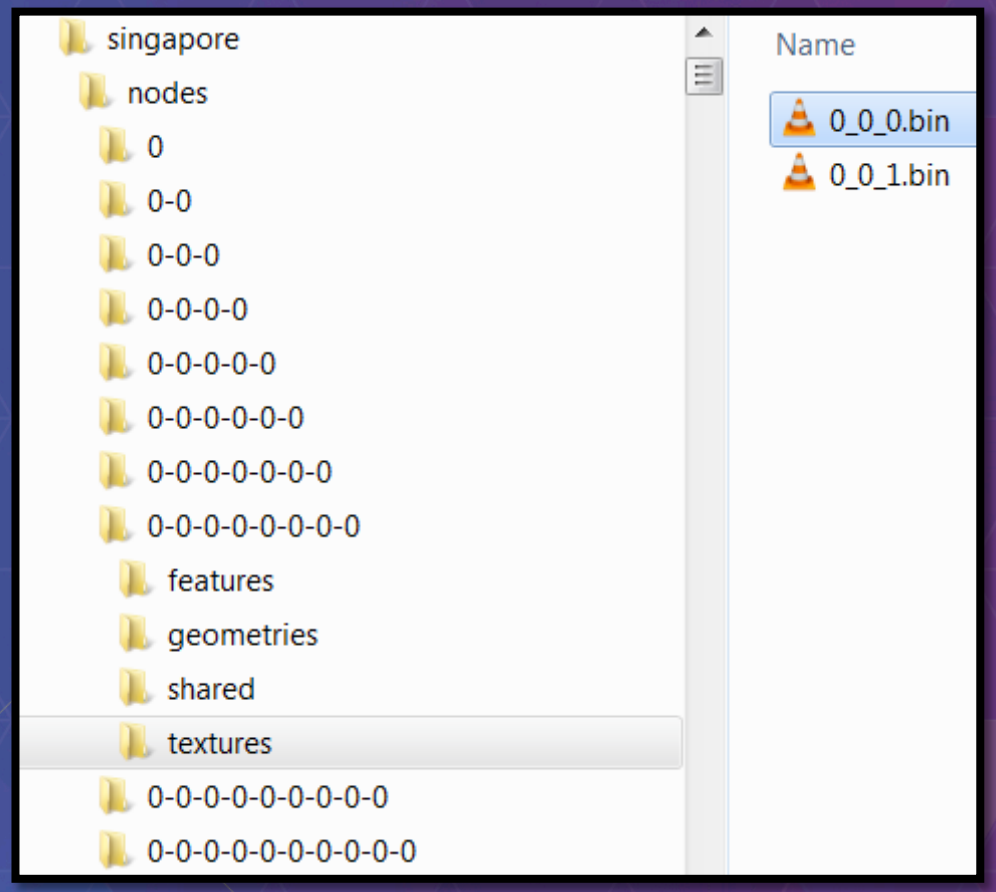
Storing and serving i3s resources

File System Folder Layout

Direct mapping of the REST API

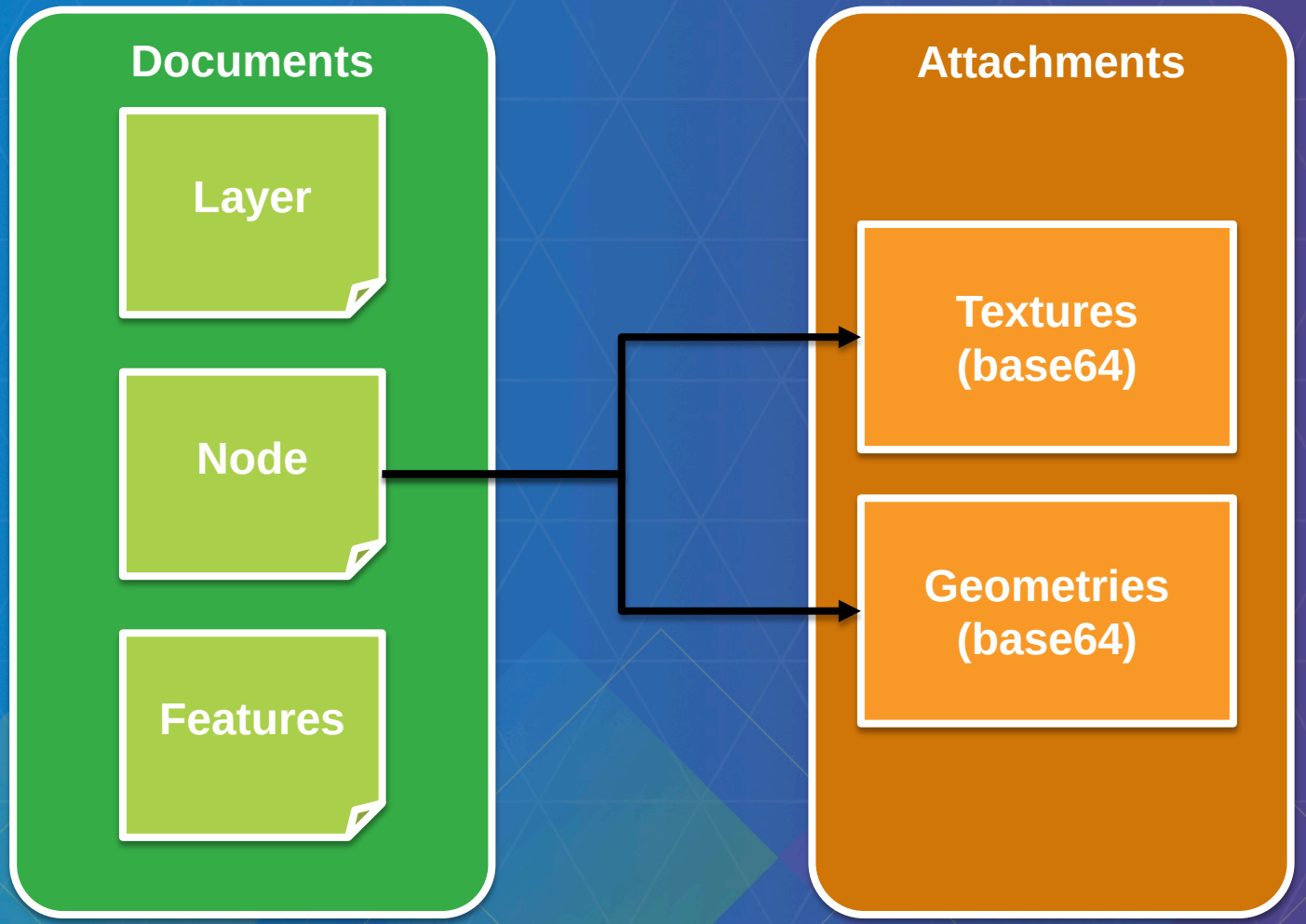
- **All resources reside in the file system as individual files.**
- **These files are organised in folders in the following schema:**

```
/3dSceneLayer.json
/nodes/root/3dNodeIndexDocument.json
/nodes/root/features/0.json ...n.json
/nodes/root/geometries/0.bin ...n.bin
/nodes/root/shared/SharedResource.json
/nodes/root/textures/0_0.bin ...n_m.bin
```



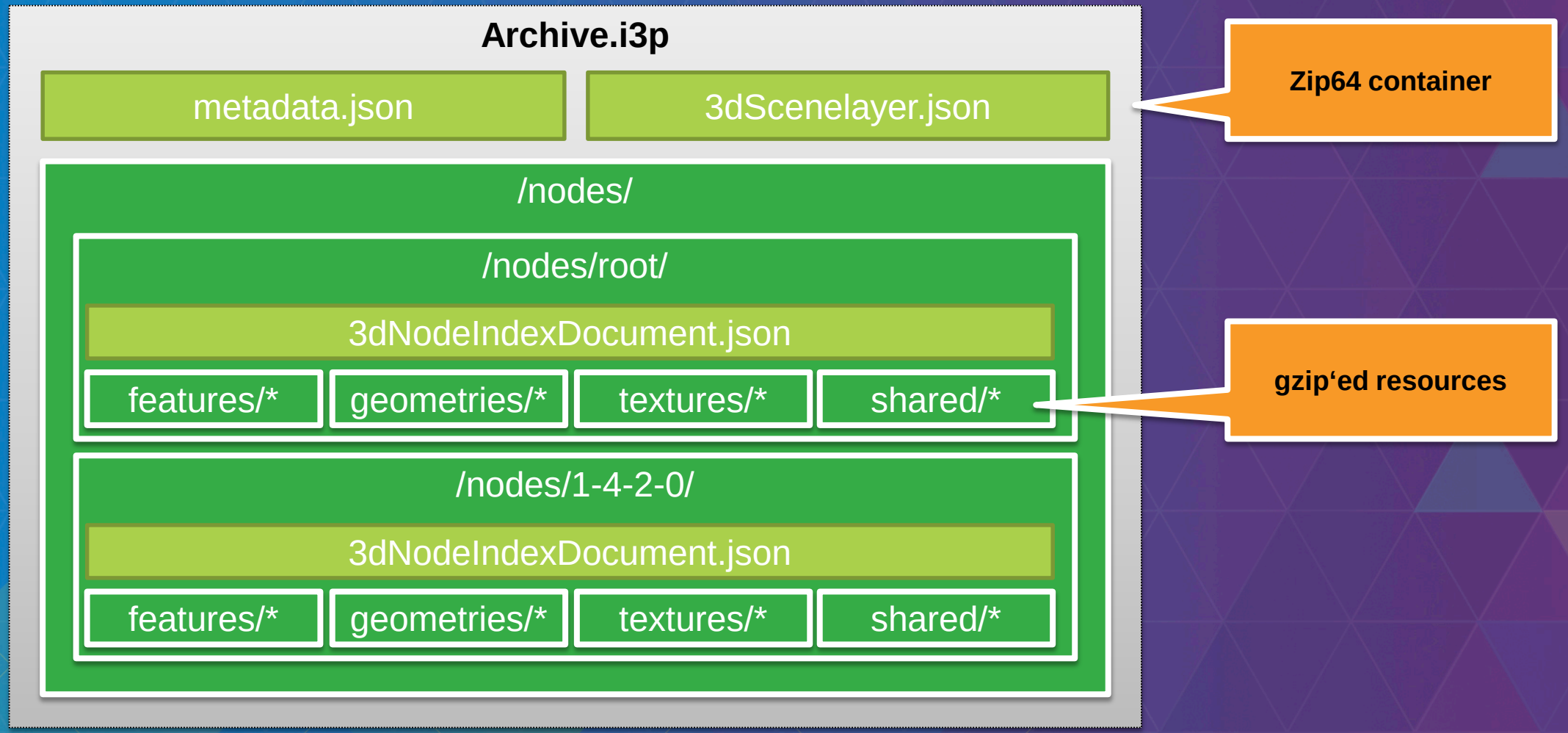
CouchDB – JSON Object Storage

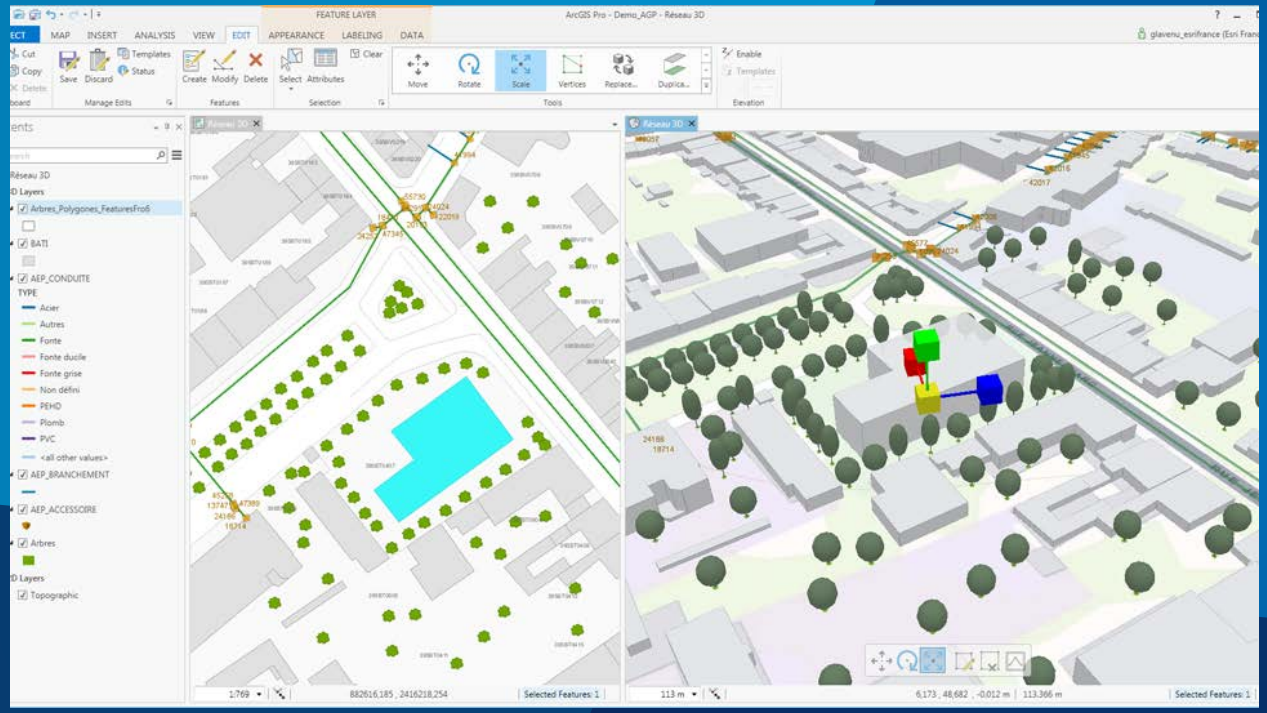
Apply liberally with IndexedDB or other KV-Stores



i3p – the i3s package format

Best for single-file exchange and portability of i3s stores





The i3s toolkit

Basic tools to build i3s-based services and apps

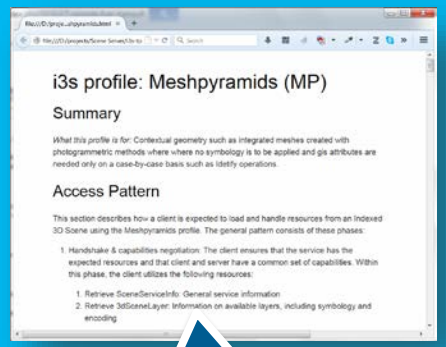
Scene Service & i3s Specifications

License: Creative Commons – Attribution – No Derivatives 3.0

Scene Service REST API Specification

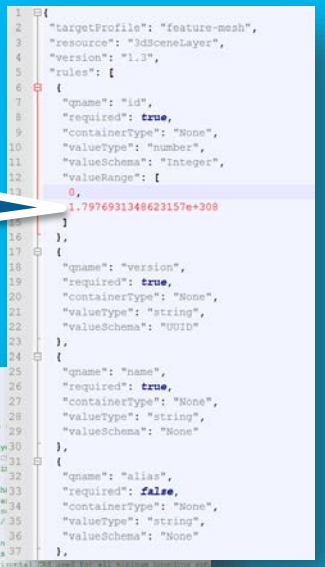
Indexed 3D Scene Format Specification

Profiles (8x)

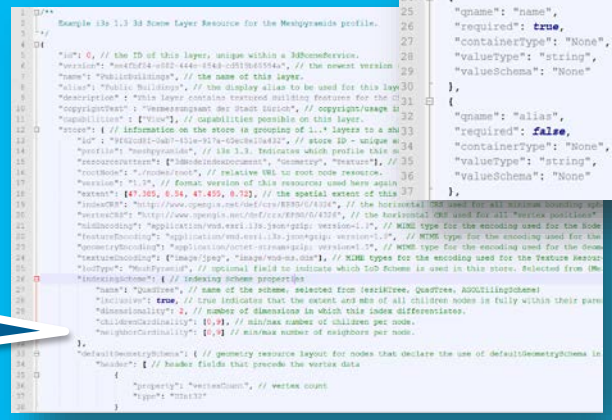


Documentation

Validation Rules



Examples



I3s Rulesets

Formal Definition of the specification

- Modularity
- Structure
  - Type
  - Pattern
  - Cardinality
  - Nullable
- Domain Restrictions
- Dependencies
  - „Apply a rule only if this set of conditions is met“

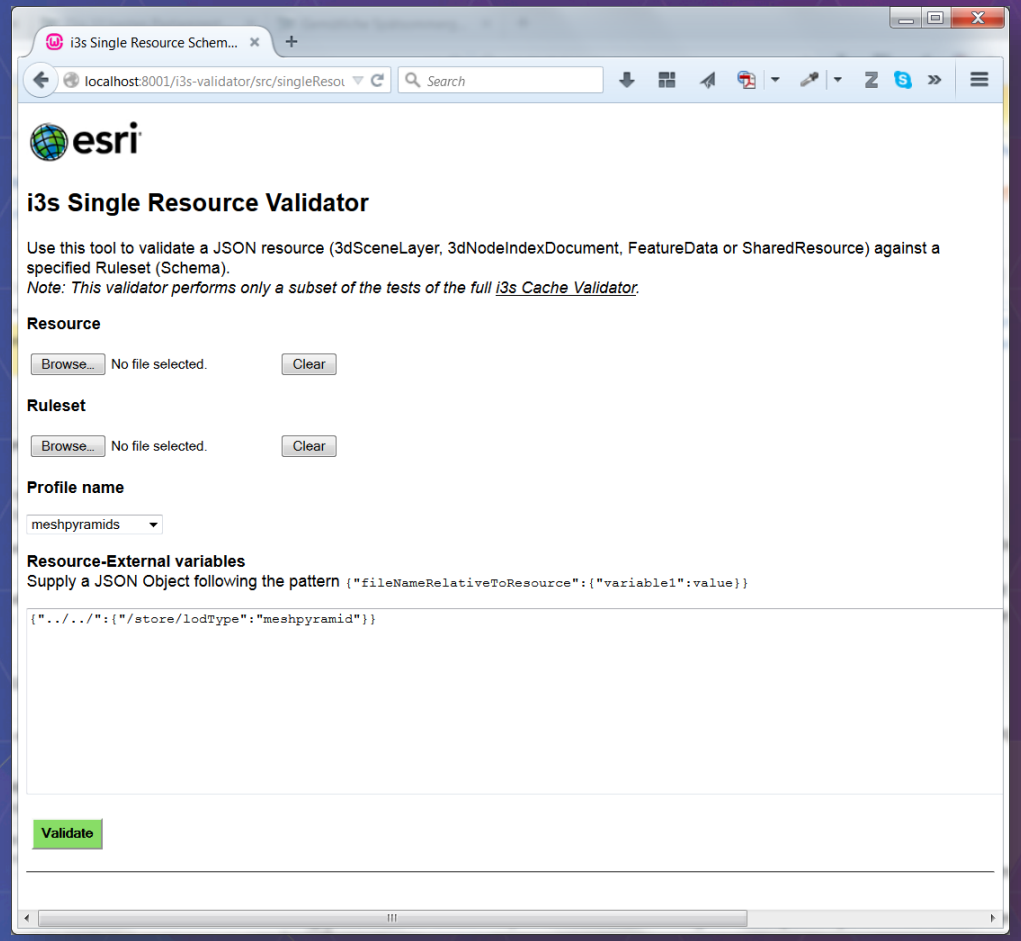
```
{
 "qname": "featureData.*.geometries.*",
 "required": true,
 "valueType": "object",
 "valueSchema": "NamedRuleset",
 "namedRuleset": "../../all/rules/GeometryArrayBufferViewRules.json",
 "valueDependencies" : [ {
      "to": "type",
      "type": "IS",
      "value": "ArrayBufferView"
 } ]
},
{
 "qname": "featureData.*.geometries.*",
 "required": true,
 "valueType": "object",
 "valueSchema": "NamedRuleset",
 "namedRuleset": "../../all/rules/GeometryReferenceRules.json",
 "valueDependencies" : [ {
      "to": "type",
      "type": "IS",
      "value": "GeometryReference"
 } ]
```

Validators

Structure + content validation for JSON documents

- **Standalone web application**
- **Node.js application**
- **Embedded into grunt build process**

- **Single Resource Validator**
  - ***Schema validation*** **of a single file**
- **Store Validator**
  - ***Consistency validation*** **of a full or partial store**

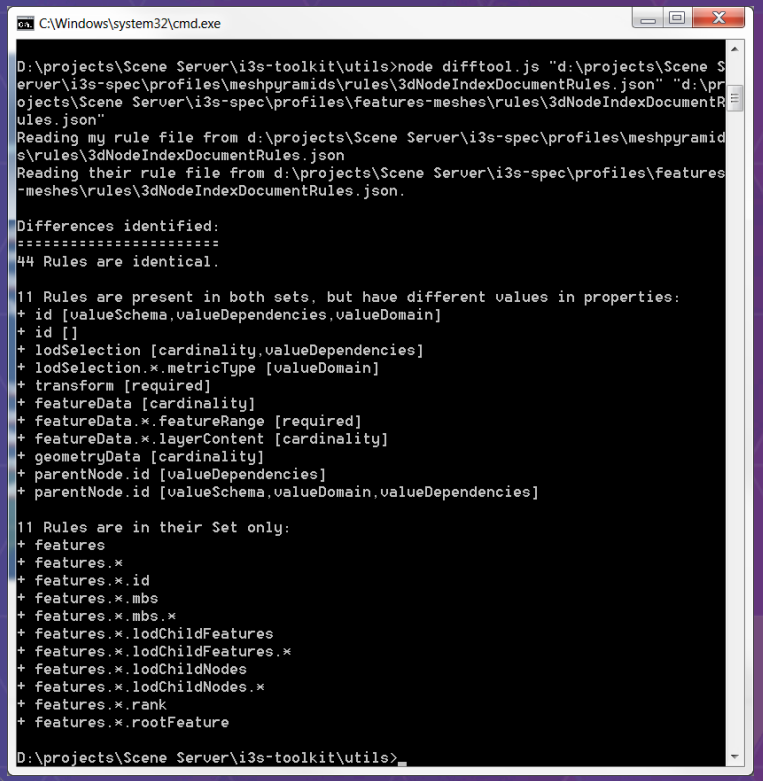


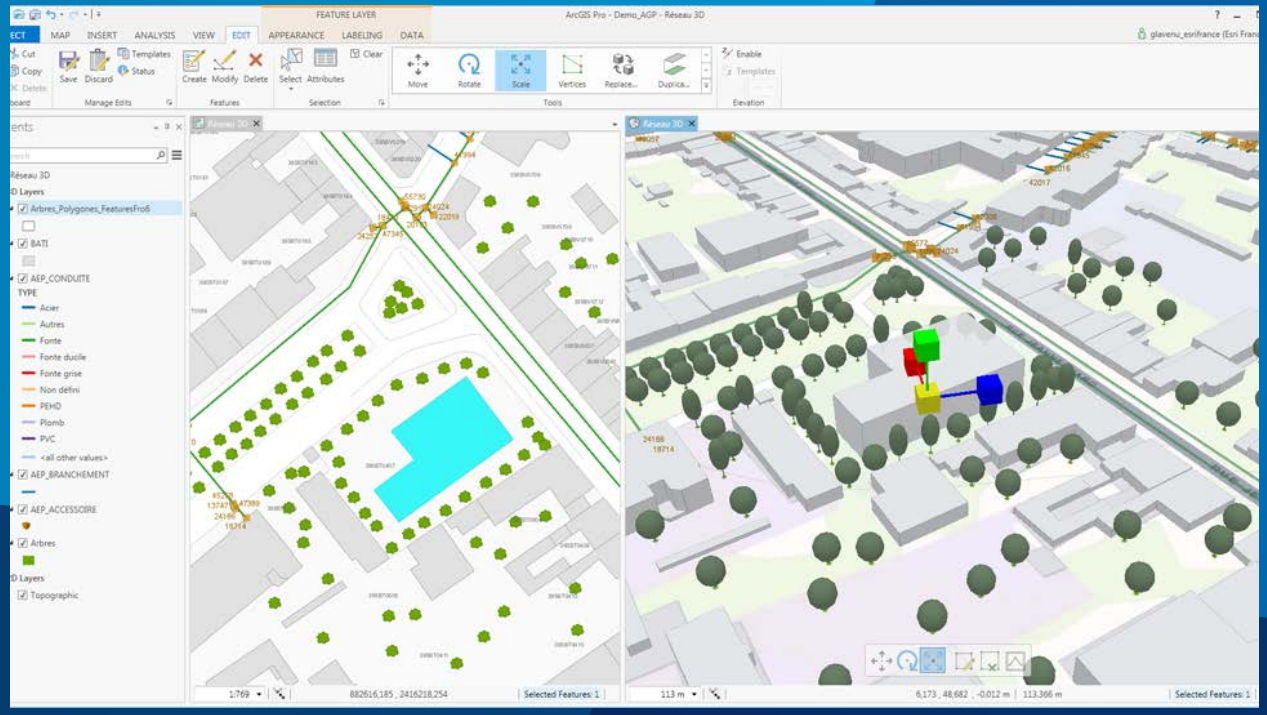
Node.js Utilities

Helpful tools to create the specification and work with i3s content

- **Working with services:**
  - `harvest.js`: Download a published Scene Service from its‘ REST API and store on disc
  - `packi3p.js`: Create an i3p file form a cache on disc
  - `unpacki3p.js`: Unpack an i3p file to disc

- **Working with i3s rulesets:**
  - `schemagen.js`: Generate a ruleset from an example
  - `multischemamerge.js`: Perform an additive ruleset merge on a folder of ruleset files
  - `difftool.js`: Find differences between two rulesets
  - `docgen.js`: HTML documentation from ruleset





Partner Implementations

How to become part of ArcGIS Web Scenes

Vision

An Open Specification for the Geospatial Community

- „The“ visualisation format for any kind of Geospatial 3D Data
  - Scalable
  - Web friendly
  - Declarative
  - Full feature data
  - Visualisation friendly
- i3s satisfies a basic infrastructure need of the GIS community
- Most valuable if implemented everywhere
- Lower possible hurdles to implementation

Roadmap for working with Partners

Current & planned process

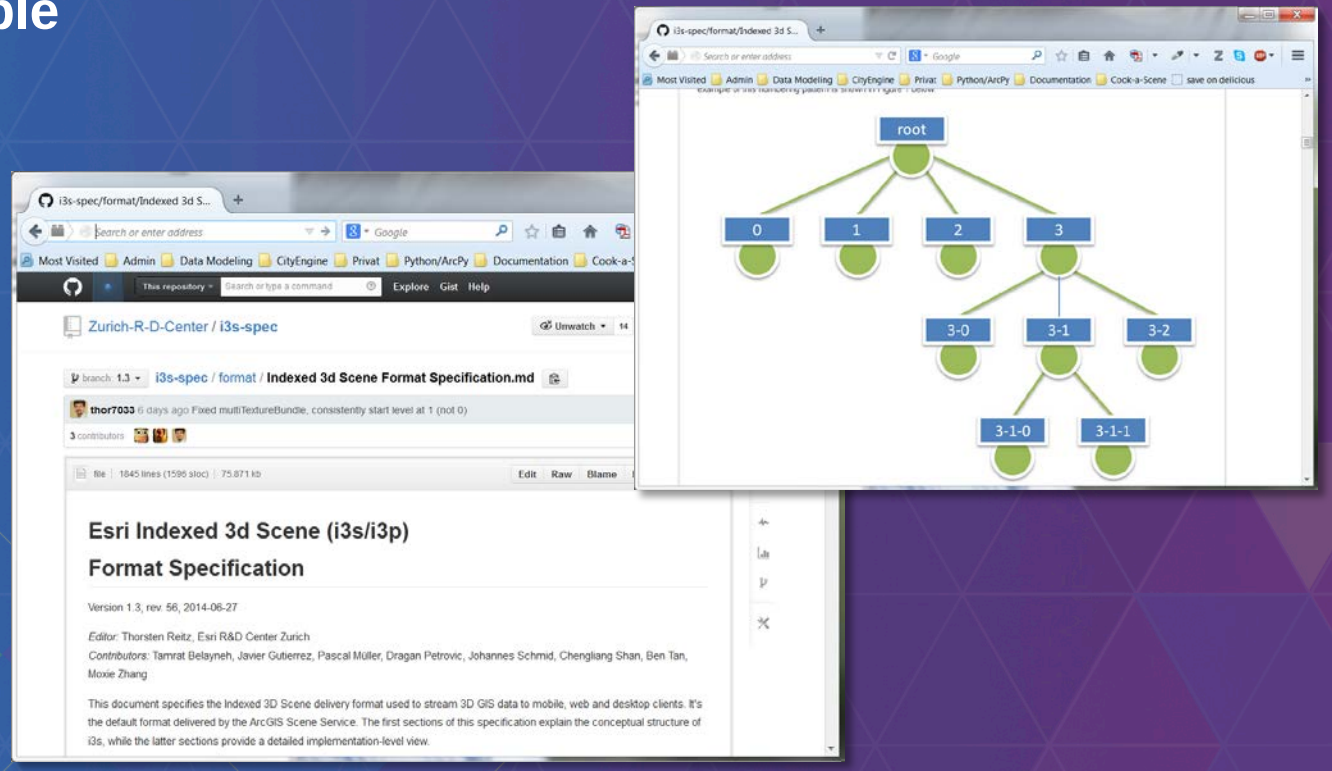
- **Phase 1**
  - Rapid spec evolution with small set of partners
  - Performance Optimization
  - Clean Up
- **Phase 2**
  - Slower spec evolution with full public participation
  - New Format Features

- **Collaboration through Github**
  - „caniuse“ of features of each known implementation

Get Involved with Scene Services!

Implement Encoders, Services, Clients

- **Specification License:** Creative Commons-Attribution-No Derivatives 3.0
- **Toolkit License:** Apache
- **Specification and Toolkit available upon request for early adopters**
  - **Ask us at**
    - treitz@esri.com
    - candrews@esri.com

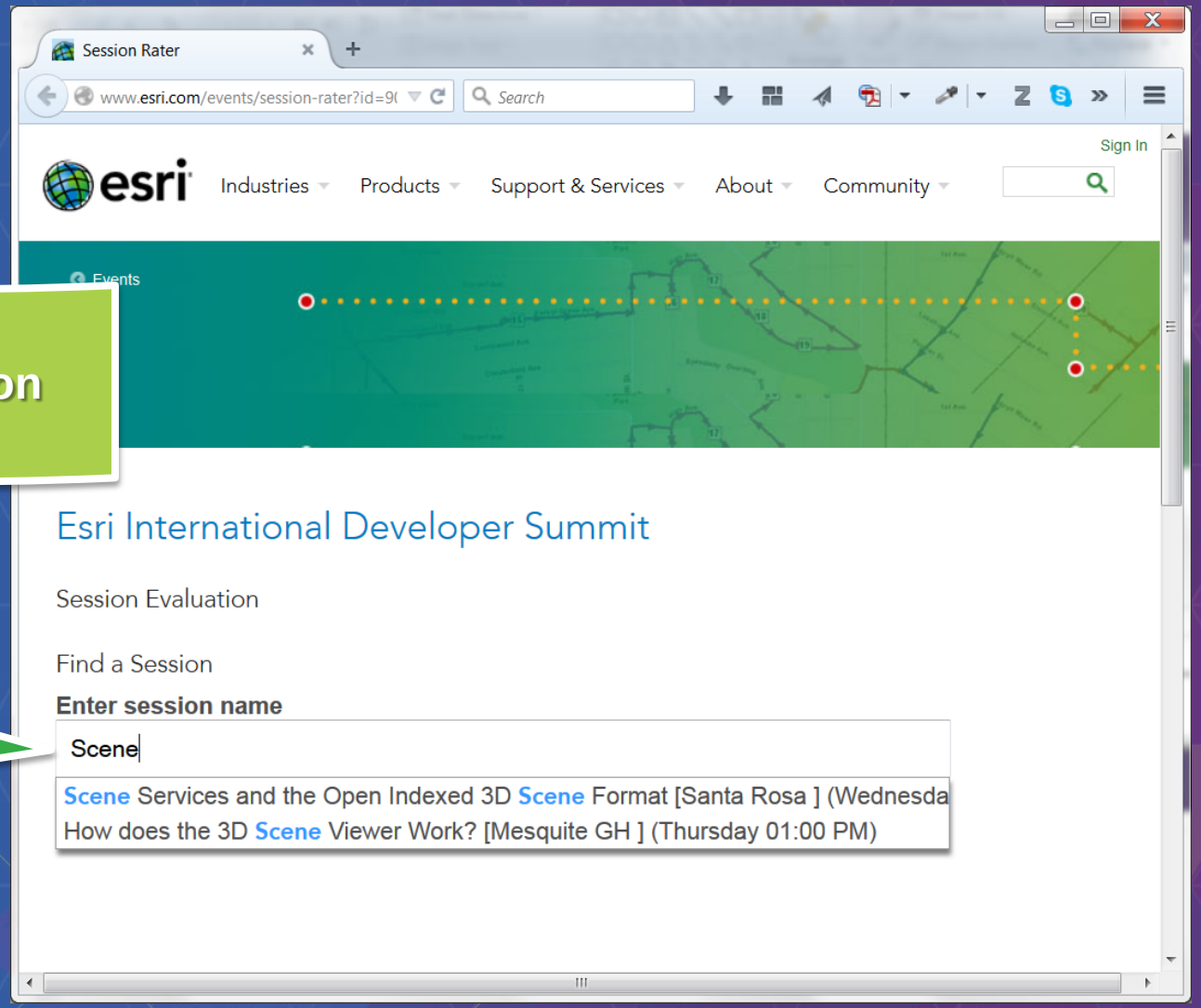


Thank you…

Any Questions? Ideas? Comments?

Esri.com/RateMyDevSummitSession

Scene



Understanding our world.