Asset Addressing

Many DSON objects have an `id` property. These objects can be addressed using the Uniform Resource Identifier (URI) fragment identifier notation.

The syntax of URIs is defined in the Internet Engineering Task Force (IETF) document RFC 3986 - "Uniform Resource Identifier (URI): Generic Syntax". It defines a URI as consisting of five hierarchical parts: the scheme, authority, path, query, and fragment.

The syntax is:

```
URI         = scheme "::" hier-part [ "?" query ] [ "#" fragment ]

hier-part   = "//" authority path-abempty
             / path-absolute
             / path-rootless
             / path-empty
```

The scheme and the hierarchy-part are required. The hierarchy-part, however, can be an empty path. URI syntax requires that the hierarchical path name levels (such as directories) be separated with forward slashes ("/") and that other special characters within the path be escaped, for example, converting spaces to their hexadecimal representation “%20”. Entries are all considered case-sensitive.

In the DSON format, absolute path names (i.e. path names that begin with a leading "/") are understood to be relative to the content root folder rather than being relative to a particular drive or file system root directory. Absolute path names that include directories above the content root folder (e.g. “/C:/My Documents/My Content/Victoria/BaseMorphs.dsf”) are discouraged in the format to allow assets to be addressed within a contained content directory. An absolute path that does not conform to the IETF convention must be adjusted to do so. For example, a Windows path “\foo\bar\my file#GirlMorph.dsf”, by URI syntax definition, contains backslashes that could be treated the same as any other text character, not as valid separators. Although some applications look for Windows paths and convert them to valid URIs, not all applications do. Therefore, always use valid URI syntax, which for this example would be “/foo/bar/my%20file%23GirlMorph.dsf”.

Assets within a file are addressed using the fragment identifier (“#”). In a URL property, that is when an asset is to be looked up based on a URI, the URI fragment identifier is preceded with the literal pound sign or hash character (“#”). When defining the `id` for an asset, no hash symbol is used, and the `id` should not contain any path information, as this is derived from the file’s URL in combination with the asset’s path within the *_library in the file.

Whenever possible, it is better encoding practice to use paths that are relative to the location of the document that references them rather than to use absolute paths.

All `id` values within a given file must be unique, regardless of what Object Definitions they are part of.
URI Schemes

A URI may be preceded by one of two named schemes to indicate whether the instance reference (see below) is stated in ID’s or in names. The two legal schemes are id and name. An id scheme means that the object reference uses ID’s (id attribute) for each component of the path. A name scheme means that each component in the path uses a (potentially non-unique) name (name attribute). If no scheme is given, id is assumed.

Object References

Many elements need to refer to other objects that may or may not already be in the scene when attempting to load a file. In the scene element, the ordering of nodes, modifiers, geometries, materials, formulas, and settings is important because it allows a scene to be built up as a parser traverses each of these elements in order.

An object reference is composed using URI syntax where the root URL is the node path within the scene along with an optional object type selector, and the query string selects what asset on the node to select. The segment selector determines what property of the asset to address. An optional sub-property selector can be appended to select a sub-component of the property.

```
[scheme://]<node_path>:<file_path>#<asset_id>[?<property_path>]
```

A property that is associated with the named asset can be selected using a property path (e.g. to select the x component of the translation property) (see below).

Example

```
"hips:morphs/Daphne#daphne?value"
```

This URI searches on the “hips” node first, to locate the asset with the URI “morphs/Daphne#daphne”. The modifier ‘daphne’ is selected, and the property on ‘daphne’ named ‘value’ is selected. If a matching element in the scene is not found, the asset “morphs/Daphne#daphne” is loaded from file.
Instance References

Within a file there are many places where object references appear. Object references may refer to items that exist within a scene, asset definitions within the current file or another file, or asset instances within the current file. Asset instance references may only be used to refer to instances within the same file (i.e. instances created in the scene element of the file). Since asset instance references only refer to instances within the file, they should all begin with a ‘#’.

Node Path

A node path is the full or partial path to a node in the scene. If the path starts with a leading slash (‘/’), searching begins at the scene root node(s). The presence of a slash (‘/’) tells the search algorithm to look for any descendent of a node using a breadth-first search. For example, for the path ‘/hips/rfoot’ the search algorithm should start searching at the scene root node ‘hips’, then find the first descendent of ‘hips’ with the id ‘rfoot’.

If the node path does not begin with a leading slash, but rather starts with the @selection tag, then the application should search within the current scene selection scope first to locate the first node or modifier in the path. If not found in selection scope, then continue the search with all root nodes in the scene, traversing the node hierarchy until the top-level node in the path is found. Path search should follow normally from there.

Example

"/scene/rfoot:figures/Victor#rfoot?translation"

This path starts at the scene root and looks for the first node instance named ‘rfoot’. Once located, it selects the property on ‘rfoot’ named ‘translation’. If the node does not exist in the scene, the application can choose to halt loading, or it can retrieve “figures/Victor#rfoot” from file.

"hips:figures/Daphne#iliac_crest?value"

This path selects the first instance of ‘hips’ found by first traversing the current selection context, then by traversing the scene. The modifier ‘iliac_crest’ is selected if found, and its property named ‘value’ is selected.

Property Path

A property path is a slash (‘/’) delimited path that allows selection of sub-components of assets. It follows similar rules to Node Paths in that a slash (‘/’) may be used to request a search of all descendent properties and sub-properties rather than specifying every parent property in the property hierarchy.
Example

"lhand/ball: props/SportsBalls#baseball_mat?materials/stitches/color"

This URI locates the asset “props/SportsBalls#baseball_mat” (which happens to be a material) either in the scene (by searching on the lhand/ball node) or through its file reference. It then selects the property “materials” then searches for the “stitches” sub-property that is directly owned by the “materials” property. Once the “stitches” sub-property (which in practice would likely be a material instance in the scene) is located, all of its sub-properties would be searched to locate a “color” sub-property. Only the first instance encountered would be selected.

Channel Naming

Most channels may be named arbitrarily, but channels that deal with transform information must be named according to the transform axis that they relate to. “x”, “y”, and “z” are the channel names that must be used for any transform channel that has an axis dependency.