Autodesk®

Maya® 2009 software

Maya Assets

Autodesk®

Maya Assets

In Autodesk® Maya® 2009 software, Maya Assets enable you to encapsulate a set of nodes and treat them like a single node, with only the important attributes exposed.

This builds on the concept of container nodes in previous versions of Maya, where container nodes were used mostly to simplify the Hypergraph display.

Examples of the new container node capabilities:

- Tight integration with the primary editors in Maya.
- Simplification of complex hierarchies.
- Additional options for creating containers.
- Customizable definition and organization of container attributes.
- Ability to associate parenting relationship data with a container.
- Ability to transfer values, connections, and relationships between containers.

Additionally, locking support and file referencing integration let you create containers that can be referenced and treated as a 'black box' by other users.

The following sections explain these concepts in more detail.

Containers and Published Attributes

Think of a container as a set of other nodes. The contents of one container cannot overlap with another container, but the containers can be nested. [Note: Nodes inside of a container must originate from the same file or be referenced containers.]

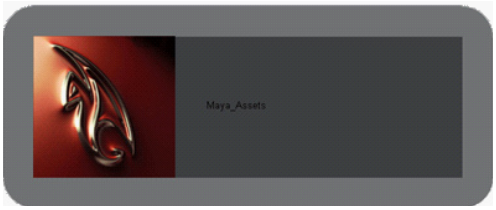

Figure 1. Maya container with custom icon.

Any attribute on a contained node can be published to the interface of the container with a user-defined name. This relationship is stored by a connection from the published attribute to the container. This connection is only used for tracking. [Note: This differs from Autodesk® Maya® 2008 software where published attributes were dynamic attributes connected to published attributes as drivers. Maya 2008 files are converted to the new architecture during file read].

The traditional method of creating a customized attribute interface involved adding dynamic attributes to a control node, then wiring the dynamic attribute into the related attribute. The container/published attribute architecture provides a number of advantages:

- The published attribute can still be manipulated directly.
- Improved performance with the elimination of the extra connection, avoiding unnecessary dirty evaluation.
- No more duplication of data.

Interaction with Commands

All Maya commands that operate on attributes accept either the original attribute name or the published name as an argument, so you can achieve the same result if you perform an operation like "setKeyframe" on the original attribute name or the published name.

Interaction with File Referencing

File referencing now uses the published names for reference edits rather than the original attribute names. This allows published attributes to tolerate name changes, unlike standard file referencing architecture. Beyond name changes, you can now change or rewire the entire contents of the asset without breaking the file reference, as long as the published attributes are still meaningful for the newly wired asset.

You can export containers in your scene as references. File referencing allows you to modularize a parent scene into multiple referenced files so that different people can work on different aspects of it without having to access the parent scene itself. Referenced objects in the main scene update automatically as changes are made to its reference files.

Although you can edit a referenced object encapsulated by a container and save those edits to the referenced file, the real power of containers and referencing comes from the fact that changes to published attributes are stored according to their published names. These reference edits are saved to the parent file instead of the referenced file.

This means that you can replace references in your scene and, so long as the container names and attributes match the previous container, your edits will apply to the new container.

Reference edits in the parent file also allows other artists to modify the contents of the containers in the referenced files without affecting the behavior of the parent file.

Container Proxy
Although you can apply a regular proxy reference to a referenced container, this greatly reduces your ability to interact with the container since its published attributes are not available. To remedy this problem you can export a referenced container as a proxy container instead.

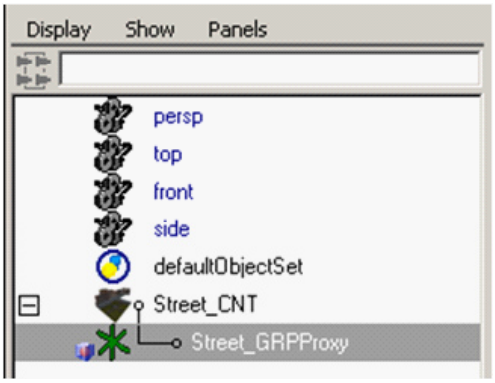

Figure 2. Street container with Street_GRP published root transform proxy.

A proxy container, like a proxy reference, allows you to substitute a potentially complex referenced container for a simpler file. Proxy containers are useful for simplifying complex scenes and help to improve navigation or performance when you are primarily concerned with the behavior of components in your scene.

When you create a proxy container, the proxy file consists of a locator representing the container's root node and an additional locator for each parent or child anchor for the container. You can add additional geometry in the proxy file to better represent the geometry it is being substituted for. Proxy containers retain published attribute values and connections even when reloaded with their proxy files in the main scene. Also, any published parent or child anchors will still appear in the scene's hierarchy so you can move the encapsulated objects around as needed.

User Interface (UI) Integration
Containers and published attributes are tightly integrated into the primary windows and editors in Maya, specifically:

Attribute Editor
Published attributes display in the Attribute Editor with the appropriate widgets for their type. For example, a published color attribute displays with a color-wheel and color-slider. Numeric attributes display with sliders and appropriate minimum and maximum values.

By default, all containers have two view modes: flat and node-based. These can be set individually per container. Flat mode lists the attributes in the order they are published and node-based groups publish attributes into collapsible frames with one frame per node. Additionally, you can define custom grouping and ordering by setting the view mode to 'template' and creating custom views. (See the Template section.)

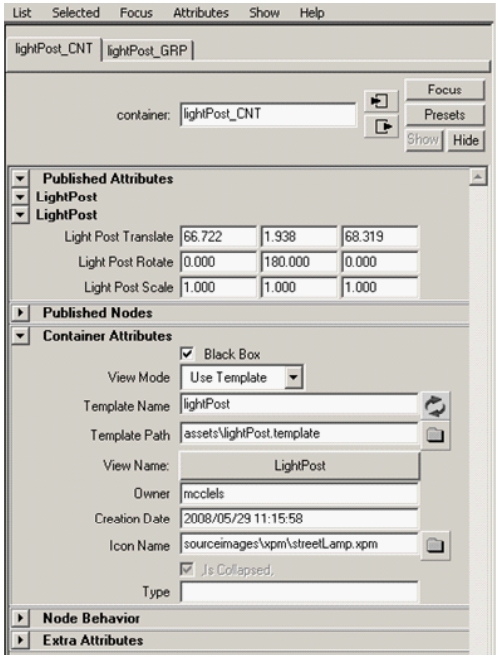

Figure 3. Attribute Editor display of lightPost container showing container and published attributes.

Channel Box
The Channel Box uses the viewMode for ordering published attributes in the same manner as the Attribute Editor (with the restriction that the Channel Box shows only numeric attributes and attributes set as keyable or displayable).

The Channel Box also has a new Show menu that allows attribute filtering on all nodes (not just containers). An additional attribute filter (Published) lets you restrict the display to only published attributes.

The Containers option in the Show menu lets you limit the Channel Box to display only container attributes for selected objects. You can also choose to display the container first in the Channel Box when any container member is selected. When an attribute in the container is highlighted in the Channel Box, the related node is selected in the scene, and displays with the appropriate manipulator for translate/rotate/scale.

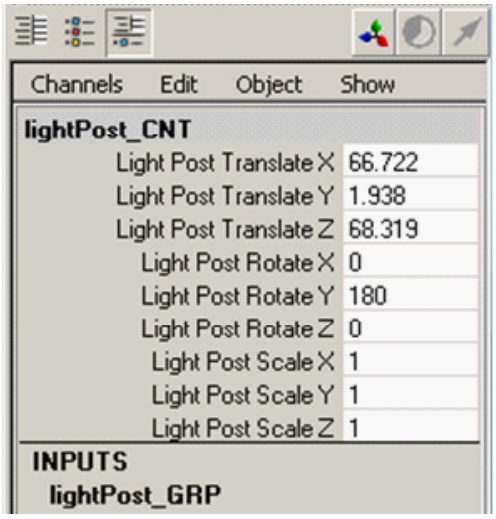

Figure 4. Channel Box display of lightPost container's published attributes.

Autodesk®

Outliner
New options in the Outliner let you display a container-centric view of your scene, with nodes organized under their related container. You can also see a published hierarchy view in which only nodes that are published as Root Transform, Parent Anchor, or Child Anchor are shown. You can drag and drop to edit container membership or to reorder the nodes under a container.

You can also view published attributes under their container in the Outliner. Attributes are ordered as specified by the container's view mode and obey the attribute filters.

Assets also accept custom icons which (when using a 20x20 XPM image file) are drawn as the container's node icon in the Outliner. This custom icon is also used as the Published Root Transform's node icon to help distinguish the asset's hierarchy from a regular Maya hierarchy and from other asset hierarchies.

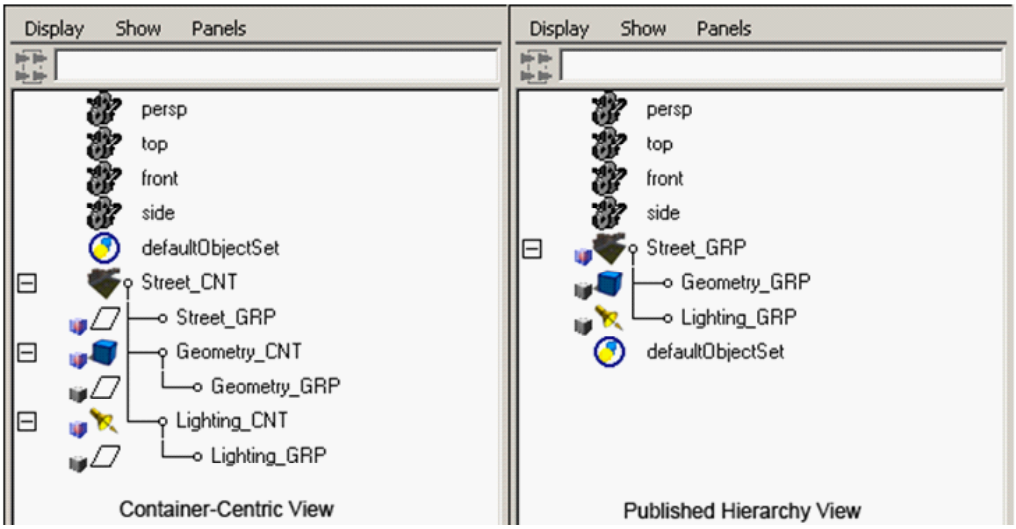

Figure 5. The difference between Container-centric display and published hierarchy display in the Outliner.

Hypergraph and Hypershade
New Hypergraph/Hypershade features for containers include:

- Associate a background image with a container and an icon with a collapsed container.
- Drag and drop to perform connections between containers.
- Major performance improvements.
- Black Box mode, which prevents expanding a container in the Connections view of the Hypergraph and culls the display of the asset's non-published nodes in the Hierarchy view.

The following enhancements apply to both containers and non-containers:

- Ability to hide relationship connections (connections where data does not flow such as between set nodes and their members).
- Ability to collapse multiple connections between nodes into a "fat connection".

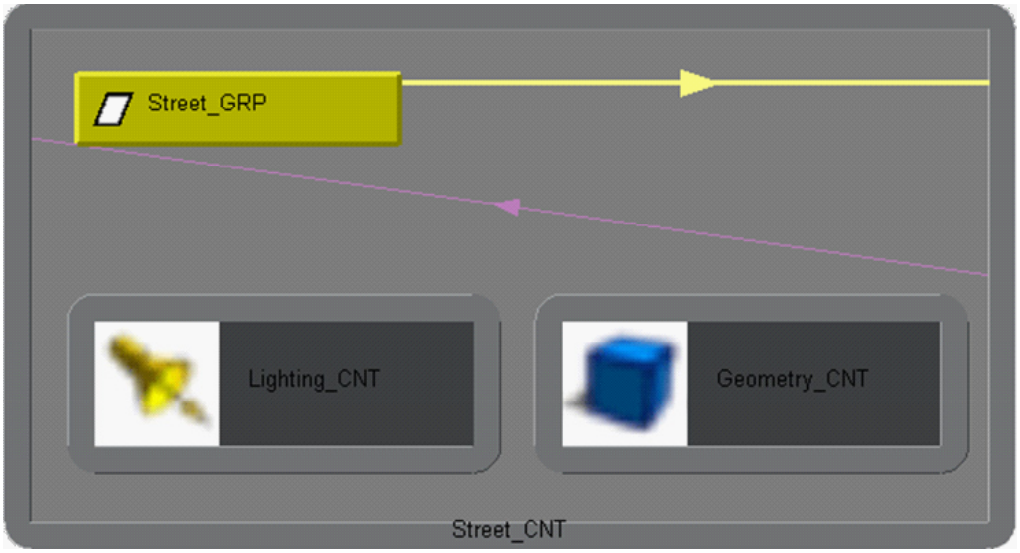

Figure 6. Hypergraph display of nested containers and Merge Connections of published attributes.

Asset Editor
A new Asset Editor window lets you accomplish asset-related tasks:

- Publishing/unpublishing Parent and Child Anchors and Root Transform nodes.
- Publishing/unpublishing attributes.
- Publishing aliased attributes.
- Binding/unbinding published attributes.
- Re-organizing nodes in a container.
- Defining a template and views for an asset.

High-level Relationship Data
In addition to attributes, certain kinds of relationship data can be published to the container. The motivation for publishing such relationships is two-fold: it clearly defines the allowed interface with external nodes and it allows higher-level operations to understand how to work with the container. It also allows container commands to transfer such relationships between containers and to fully support them during file referencing.

The first supported relationship is a parent and child hierarchy. By publishing a node as a Parent Anchor, you specify that the node is allowed to serve as a parent to nodes outside the container. Similarly, a Child Anchor denotes that a given node in a container can be parented to external nodes. Any number of nodes can be published as parent/child anchors with user-defined names to differentiate them.

The second supported relationship is a published root transform. By publishing a node as a Root Transform, you specify that the node is allowed to serve as a parent and a child to nodes outside the container. Furthermore, you can define a selection priority within the viewport when using Container-Centric Selection, which uses the asset's Root Transform as the first priority when selecting an object associated with the asset.

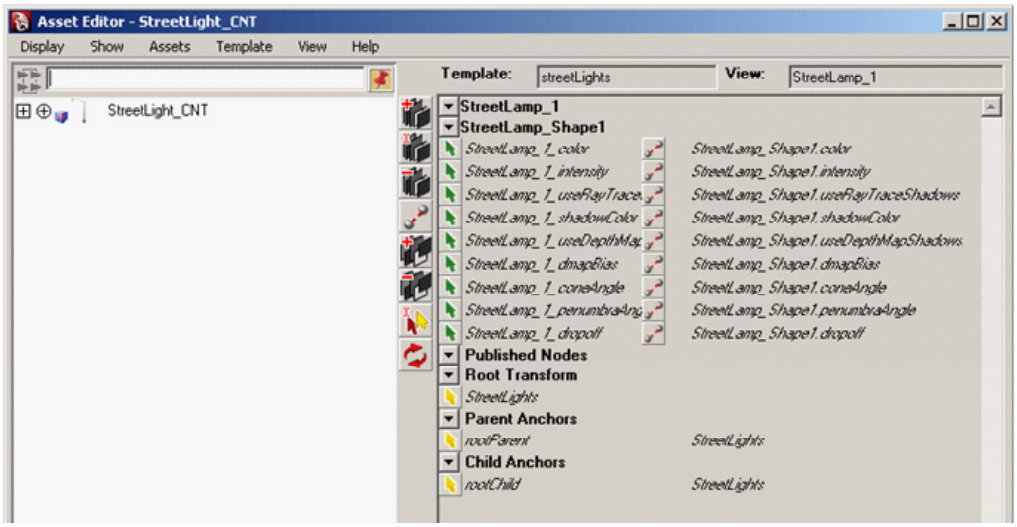

Figure 7. Asset Editor display of Street Light container and published interface.

As with standard published attributes, file referencing and 'copy attribute values' understand the published parenting relationships and know how to swap in the new parent or child (regardless of its name). For example, when creating a character that holds 5 props, put each prop in its own container. Publish the top transform on each as a parent with the name "HoldMe" and save these as separate files. Reference one prop file into the scene and parent the prop to the character's hand, then use 'Replace reference' to swap in any of the other props.

Current Container
Newly created nodes are automatically placed in the current container. You can set any unlocked container to be the current container, letting you fill a container on the fly. For example, if you create a container and make it current, then build a shader, all the new shader nodes go into the current container automatically. In this respect, containers can be thought of like visual namespaces.

Black Box

A 'Black box' attribute has been added to containers. Enabling this attribute means:

- Only published nodes appear in the Outliner, Hypergraph (both DAG and Dependency Graph views).
- Container-centric selection is enforced regardless of the container-centric preference you set, so you can only select published nodes.
- Pickwalking only allows walking to published nodes.
- Containers cannot be expanded in the Hypergraph.
- The Dependency Graph Traversal tool doesn't allow navigation to a non-published node.
- The Attribute Editor navigator buttons don't allow navigating to a non-published node.

Locking

Maya 2009 adds the ability to select a container and lock its unpublished attributes. Locking unpublished attributes does the following:

- Locks the container node.
- Locks the nodes within the container.
- Disallows modification of unpublished attribute values.
- Prevents making or breaking connections to unpublished attributes.

When a container is locked, the standard node locking rules apply. In addition, you cannot:

- Add or remove nodes from the container.
- Unlock member nodes inside the container.
- Publish or unpublish attributes.

When a file with locked containers is referenced, its locking state cannot be modified from the parent file.

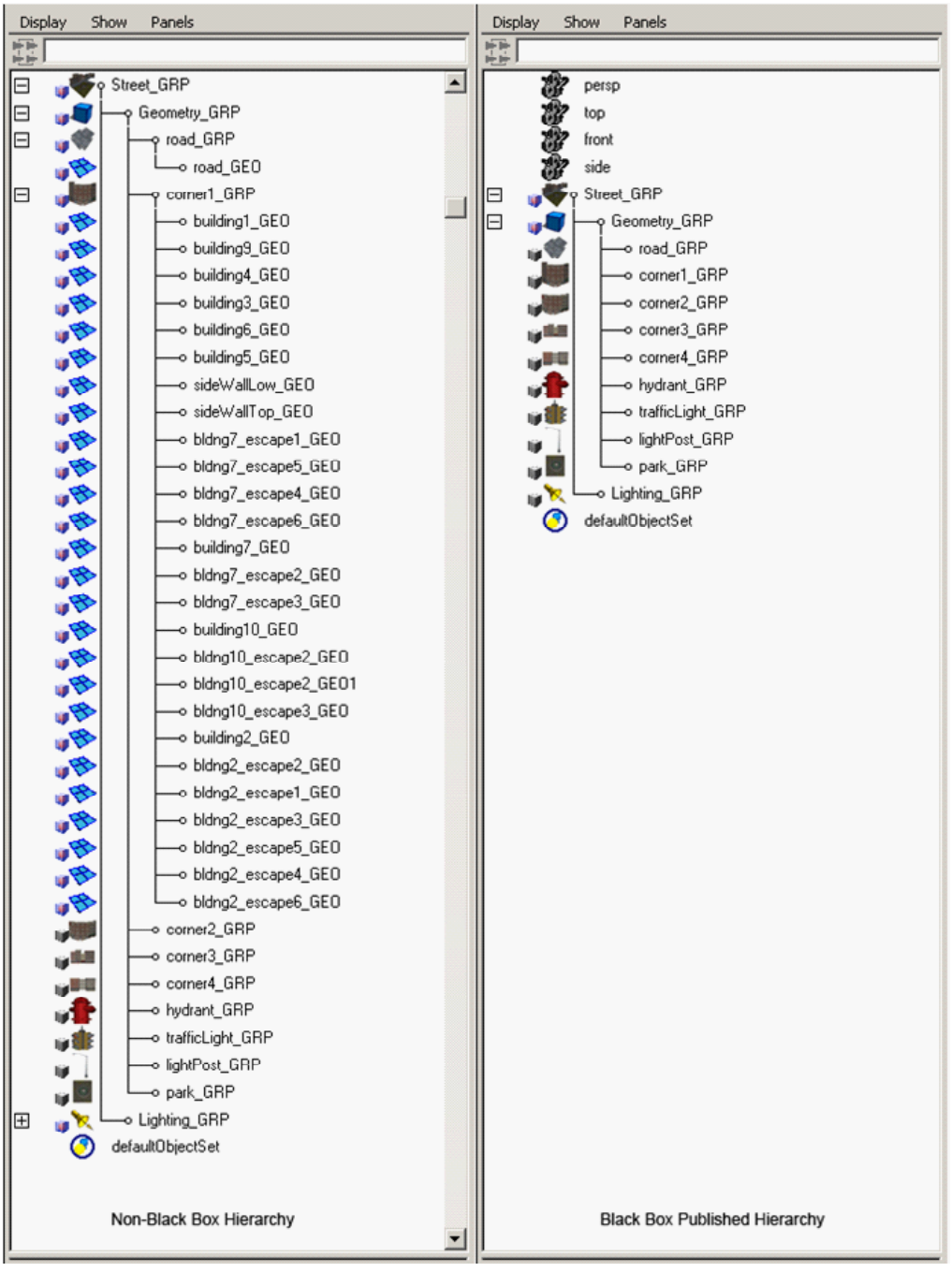

Figure 8. Glass Box container display in the Outliner versus the Black Box container display.

Templates
A container template can be used to define the user interface of the asset and to specify different views for different users of that interface.

The template file format allows hierarchical template definition with inheritance from multiple types, similar to C++ style classes. The template can be used to pre-populate a container with the desired interface as well as to validate its structure.

For example, to create a set of assets with the same interface (so their controls/animation can be easily swapped) you can create a container and publish its attributes, then export a template and use it to pre-populate the other containers with the same interface.

The container interface is saved with the node itself, so usage of templates is optional. Because it is a separate file, you can create a library of asset templates independent of specific scene files.

The other purpose of the template is to specify the UI organization of container attributes. You can define the grouping and ordering of attributes to be used in the Outliner, Attribute Editor, Asset Editor, and Channel Box. Each template can have multiple views, so users can switch between different views depending on their task.

Templates can be nested into packages for easier storing on disk, and so multiple templates can be bundled into a single file for an asset.
Maya uses an XML format for templates, which stores published attributes, views, nested templates as well as labels (aliases) for published attributes and data types for published attributes.

```
<?xml version='1.0' encoding='UTF-8'?>
<templates>
    <using package='maya'/>
    <template name='ramp'>
        <attribute name='colorRange0' type='maya.TdataCompound'>
            <label>Color Range 0</label>
        </attribute>
        <attribute name='colorRange1' type='maya.TdataCompound'>
            <label>Color Range 1</label>
        </attribute>
        <attribute name='falloffCurve0' type='maya.TdataCompound'>
            <label>Falloff Curve 0</label>
        </attribute>
        <attribute name='falloffCurve1' type='maya.TdataCompound'>
            <label>Falloff Curve 1</label>
        </attribute>
        <attribute name='falloffCurve3' type='maya.TdataCompound'>
            <label>Falloff Curve 3</label>
        </attribute>
        <attribute name='falloffCurve' type='maya.TdataCompound'>
            <label>Falloff Curve</label>
        </attribute>
    </template>
    <view name='viewName' template='ramp'>
        <property name='colorRange0'/>
        <property name='colorRange1'/>
        <property name='falloffCurve0'/>
        <property name='falloffCurve1'/>
        <property name='falloffCurve3'/>
        <property name='falloffCurve'/>
    </view>
</templates>
```

Figure 9. Asset Template XML structure and syntax.

Container Creation Techniques
As in previous versions, you can simply select which nodes you want in a container. New creation options let you decide whether to include children, parents, upstream connections, and downstream connections.

Transfer of Data Between Containers
The new copyAttr command and its corresponding menu item (Edit > Transfer Attribute Values) let you transfer external connections, published attributes values, and relationships between nodes. The transfer is based on attribute names. This capability was originally implemented for containers based on published names, but has been extended to work on arbitrary nodes. You can specify whether the copy is performed on all matching attributes or only on attributes selected in the Channel Box or specified by command line.

As with the file referencing enhancements for containers, this capability means you can treat the container as a 'black box' when working external to it.

An example workflow: A template is defined for an asset and five variations on this asset are created, all conforming to the template. One user may publish the "width" as the scale of pCube1 and one may publish the "width" as the radius of pSphere3. Another user can then apply his animation/settings from one asset to all the others.

Export Containers: Build an Asset Library
You can build a customized library of assets by exporting container nodes. When a container node is exported, its contents, its creation information, the original name of the asset file, and custom notes are also exported.

Application Programming Interface (API)
The container API consists of the classes MFnContainer and MContainerMessage and contains the following functions:

- Create a container containing a specified list of nodes
- Query the nodes in a container
- Traverse the container hierarchy of subcontainers
- Query the published attributes on a container
- Publish an attribute to a container
- Register a callback when attributes are bound or published