Assetlib Shaper Usage

Usage Considerations

Assetlib Shaper uses as main input an asset produced by SceneComposer.

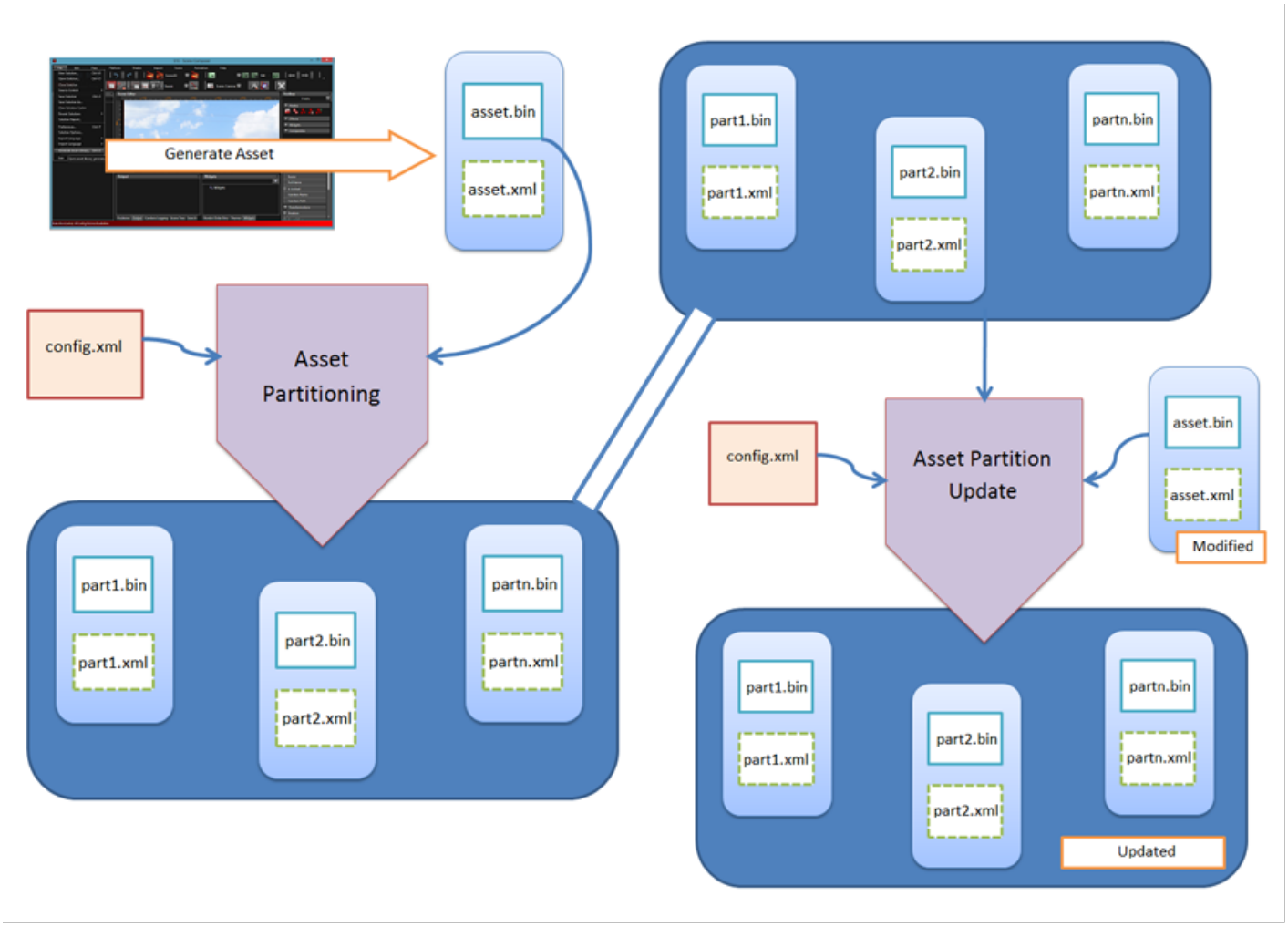
Besides the asset file, during the asset generation, SceneComposer also creates the asset report file at the same location and with the same name as the asset file (only their extensions differ). It is important to keep the report file of the asset in the same location as the asset (for example, asset partitions update works only when all the asset report files are at the same location as their associated asset files).

During the partitioning process Assetlib Shaper is looking for the asset report file of the asset provided as input. If this file is not found in the asset location, the partitioning process will output only the partitions of the given asset, no asset report being produced for these partitions. In this case, Assetlib Shaper shows a warning because it won't be possible to further update these partitions (with the content of an updated version of the original asset).

Example: you have an asset and its asset report (produced by SceneComposer during asset generation) and the partitioning configuration file. The general workflow of Assetlib Shaper:

- Add language packs to the asset if multi-language support is required. Please note that if translatable texts are used in the SceneComposer solution, a default language pack is automatically added to the asset during asset generation.
- Run the partitioning process, obtaining the asset partitions, each partition with its own asset report. If further updates are expected, the partitioning configuration should reserve space in the asset.
- Store the produced asset partitions and their asset reports to a defined location, A.
- Further, if the SceneComposer solution is updated (bitmaps changed, elements added etc.), generate the new asset and its report and store them to the desired location, B.
- Run asset partition updating process giving as input the location A (where the currently used asset partitions are stored) the configuration stored also at location A and the new asset stored at location B. Please note that by using this updating process the changes to the original asset partitions will be on a minimum and the size of the partitions will increase only if the existing reserved space is not enough for storing new or updated objects. Assetlib Shaper allows for iterative updates.

The next image shows the partitioning and partition updating workflows:



Command Line Arguments

The Assetlib Shaper, as part of the Asset Tool, is a command line tool providing the following options:

```
AssetTool -shape -h
```

Shows shaper help

```
AssetTool -shape -c
```

Shows shaper configuration file help

```
AssetTool -shape -k assetFilePath
```

Checks the format of the asset found at 'assetFilePath'

```
AssetTool -shape -it destinationfilePath assetFilePath
```

Generates asset info found at 'assetFilePath' to the text file at 'destinationFilePath'

```
AssetTool -shape -ix destinationfilePath assetFilePath
```

Generates asset info found at 'assetFilePath' to the Microsoft Excel file at 'destinationFilePath'

```
AssetTool -shape -p configurationFilePath assetFilePath
```

Partitions the asset found at 'assetFilePath' using the partitioning rules provided by the configuration file at 'configurationFilePath'

```
AssetTool -shape -l assetFilePath [LanguageCode:CompiledLanguagePackSourceFilePath]+
```

Adds or updates compiled language pack sources as language entries with LanguageCode name in the asset

```
AssetTool -shape -r assetFilePath TypeOfObjectToRemove:NameOfObjectToRemove
```

Removes the object with type 'TypeOfObjectToRemove' (or all if *) and name 'NameOfObjectToRemove' (* are accepted) from the asset

```
AssetTool -shape -st blockSize assetFilePath destinationFilePath
```

Generates the data overview at destinationFilePath (text file) of the asset found at assetFilePath using the provided blockSize

```
AssetTool -shape -sx blockSize assetFilePath destinationFilePath
```

Generates the data overview at destinationFilePath (Excel file) of the asset found at assetFilePath using the provided blockSize

```
AssetTool -shape -u currentAssetPartitionsDirPath newAssetFilePath configurationFilePath
```

Updates the partitions of the current asset with the new/updated/removed elements of the new asset with as little changes as possible.

Configuration File

The partitioning is done based on a provided configuration xml file. This file defines the partitions and the rules to be used in the partitioning process.

Configuration file defines:

- general properties of the partitions that will be created (location, block size, space reservation, compression)
- particular properties of each specific partition (name, space reservation, compression)

- assignment rules (the target partition for asset entries)

Configuration File Format

The format of the configuration file is the one shown next:

```
<?xml version="1.0" encoding="utf-8"?>
<shaper-configuration xmlns="http://tempuri.org/ConfigSchema.xsd">
    <partitions directory="Destination-Directory-Path" [block-size="Number-KB"]
                               [reserve-space-render-shader="Number-Units"] [reserve-space-names
                               [reserve-space-tree="Number-Units"] [reserve-space-entry="Number-
                                                        [reserve-space-lib="Number-Units"] [reser
        [<compression>
            [<lib type="Asset-Lib-Type"/>]+
        </compression>]
        [<partition name="Partition-Name" [reserve-space-render-shader="Number-Units"]
                     [reserve-space-names="Number-Units"] [reserve-space-tree="Number-Units"]
                                          [master="true or false"] [reserve-space-entry="Number-
                                          [reserve-space-lib="Number-Units"] [reserve-space-asse
            [<compression>
                [<lib type="Asset-Lib-Type"/>]+
            </compression>]
            [<rule type="Asset-Entry-Type" [alias="Rule-Alias"] [item="Asset-Entry-Name"] [exclu
           [reserve-space-tree="Number-Units"] [reserve-space-entry="Number-Units"] [reserve-spa
           [compress="true or false"] [case-sensitive="true or false"] [exclude-from-partition="
        [&lt;include-rules&gt;
            [<include-rule alias="Other-Rule-Alias"/>]*
            </include-rules>]
            [&lt;exclude-rules&gt;
            [<exclude-rule alias="Other-Rule-Alias"/>]*
            </exclude-rules>]
            [<include-items>
            [<include-item item="Asset-Entry-Name"/>]*
            </include-items>]
            [<exclude-items>
            [<exclude-item item="Asset-Entry-Name"/>]*
            </exclude-items>]
        </rule>]*
        </partition>]*
    </partitions>
</shaper-configuration>
```

Some of the attributes indicated during their presentation accept wildcards. Working with wildcards involves the usage of the * character as a substitute for any number of characters.

Each tag and attribute is presented next:

`<shaper-configuration>`	This is root tag of the xml configuration file. Attributes: • **xmlns** (mandatory): its only value is "http://tempuri.org/ConfigSchema.xsd", having the purpose of using an XSD schema for validating the configuration file.
`<partitions>`	This tag appears only once and, through its attributes, it defines the general properties of the outputted partitions. Attributes: • **directory** (mandatory): path of the destination directory of the produced partitions; this path can be relative (e.g., temp/partitions) or absolute (e.g., C:/temp); the location of the asset file is considered the base directory for a relative destination directory path (e.g., for destination directory temp/partitions and asset file location C:/work, the partitions are created at C:/work/temp/partitions). • **block-size** (optional): size of an individual storage block; it's value represents the block size in KB (number followed by the "KB" suffix). • **reserve-space-render-shader** (optional): amount of memory (in KB or %) reserved at the end of the render-shader region for all partitions. • **reserve-space-names** (optional): amount of memory (in KB or %) reserved at the end of the name table for all partitions. • **reserve-space-tree** (optional): amount of memory (in KB or %) at the end of the look-up tree for each lib and for all partitions. • **reserve-space-entry** (optional): amount of memory (in KB or %) reserved at the end of the data block of each asset entry for all partitions. • **reserve-space-lib** (optional): amount of memory (in KB or %) reserved at the end of the data block of each asset lib for all partitions. • **reserve-space-asset** (optional): amount of memory (in KB or %) reserved at the end of the data block of each partition.
`<compression>`	This tag refers to the compression of partitions' data. If this tag has **<partitions>** as parent tag, the compression rules defined under it apply to all partitions outputted by the partitioning process. Otherwise, if this tag has **<partition>** as parent tag, the compression rules defined under it apply to the particular partition addressed by those rules. This tag is optional and has no attributes but **<lib>** children tags (at least one if **<compression>** tag is present). Only Bitmap, ShaderProgram, VertexBuffer, RawResource and Font entries can be compressed.

`<lib>`	This tag is expected under the **<compression>** tag and defines a compression rule: an asset lib type to which compression is applied. Attributes: • **type** (mandatory): the asset lib type (e.g., Bitmap) to which compression is applied; besides the lib type names (available in Annex 1), the user can provide * as value for this attribute, representing "all lib types".
`<partition>`	This tag refers to a specific partition and defines the particular properties of that partition through its attributes. Attributes: • **name** (mandatory): name of the partition (e.g., Bitmaps), forms the file path to which the partition will be written along with the directory attribute of the <partitions> tag (e.g. C:/temp/Bitmaps.bin). • **master** (optional, but required to be true for one partition): specifies if the partition is a master partition or not; please note that all the asset entries that are not targeted by any rule will be included in the master partition. • **reserve-space-render-shader** (optional): amount of memory (in KB or %) reserved at the end of the render-shader region of the partition. • **reserve-space-names** (optional): amount of memory (in KB or %) reserved at the end of the name table of the partition. • **reserve-space-tree** (optional): amount of memory (in KB or %) reserved at the end of the look-up tree for all the libraries of the partition. • **reserve-space-entry** (optional): amount of memory (in KB or %) reserved at the end of the data block of each asset entry of the partition. • **reserve-space-lib** (optional): amount of memory (in KB or %) reserved at the end of the data block of each asset lib of the partition. • **reserve-space-asset** (optional): amount of memory (in KB or %) reserved at the end of the data block of the partition.

`<rule>`	This tag regards a partitioning rule and is in a many-to-one relation with the partition tags. It has several attributes. Attributes: • **type** (mandatory): type of the asset library to which the rule applies. It can be * (representing all types) or a specific asset entry type (e.g., VertexBuffer). For a complete list of supported types, please consult Annex 1. • **alias** (optional): name associated to a rule that can be used for referencing the rule (see include-rule or the include-rules tag). The alias is not required to be unique per rule. • **item** (optional): name pattern of the target asset entries, this attribute accepting wildcards. If this attribute is missing and no <include-items> and <include-rules> tags are defined, all the asset entries of the rule type (see the type attribute of the <rule> tag) are included by the rule. If this attribute is missing and at least one of the <include-items> and <include-rules> tags are present, this attribute has no implicit meaning (it is ignored, being undefined). If this attribute is present and the <include-items> tag is also present, then the value of the item attribute is considered as part of the defined elements of the <include-items> tag (actually, it is implicitly appended to the included items list). In this context, please note that defining a <include-items> tag containing one <include-item> is equivalent to providing the attribute item to the rule. • **exclude** (optional): name pattern of the asset entries that are excluded by the rule, this attribute also accepting wildcards. If this attribute is missing, this attribute has no implicit meaning (it is ignored, being undefined - nothing is excluded). If this attribute is present and the <exclude-items> tag is also present, then the value of the exclude attribute is considered as part of the defined elements of the <exclude-items> tag (actually, it is implicitly appended to the excluded items list). In this context, please note that defining a <exclude-items> tag containing one <exclude-item> is equivalent to providing the attribute exclude to the rule. • **case-sensitive** (optional): the name patterns in this rule are case-sensitive (true or false; false if missing). • **exclude-from-partition** (optional): the asset entries matched by this rule are excluded from this rule's partition (true or false; false if missing). • **compress** (optional): true if the rule compresses the target asset entries or false otherwise. • **reserve-space-tree** (optional): amount of memory (in KB or %) reserved at the end of the look-up tree for the asset libraries targeted by the current rule. • **reserve-space-entry** (optional): amount of memory (in KB or %) reserved at the end of the

`<include-items>`	This tag appears no more than once under a <rule> tag and regards the definition of the asset entry name patterns of the entries that are target of the rule. This tag is optional and has no attributes, only children <include-item> tags.
`<include-item>`	This tag appears at least once under an <include-items> tag of a rule and regards the definition of one asset entry name pattern that refer to entries that are target of the rule. This tag has one mandatory attribute. Attribute: • **item** (mandatory): the name pattern of the target asset entries, this attribute accepting wildcards.
`<exclude-items>`	This tag appears no more than once under a rule tag and regards the definition of the asset entry name patterns of the entries that are excluded by the rule. This tag is optional and has no attributes, only children <exclude-item> tags.
`<exclude-item>`	This tag appears at least once under an <exclude-items> tag of a rule and regards the definition of one asset entry name pattern that refer to entries that are excluded by the rule. This tag has one mandatory attribute. Attribute: • **item** (mandatory): the name pattern of the asset entries that are excluded, this attribute accepting wildcards.

`<include-rules>`	Besides defining its own included items, a rule can include the items filtered (included minus excluded) directly by other rules, directly meaning using <include-items> and <exclude-items> tags. These rules are referenced by their alias (an alias should be assigned to rules as needed). Please note that every rule has its defined type (see type attribute of the <rule> tag). Including a rule defined for a different type by using the <include-rules> tag takes into consideration the asset entry name patterns defined through <include-items> and <exclude-items> tags and applies them to the items of the container rule type, the type of the included rule being ignored for flexibility. For example, let's consider 3 rules: alias1, alias2 and alias3. Let's say the alias1 rule includes through <include-items> tag all the asset entries having their name starting with "A" and excludes through <exclude-items> tag all the asset entries having their names ending in "A". The alias2 rule includes through <include-items> tag all the asset entries having their name starting with "B" and excludes through <exclude-items> tag all the asset entries having their names ending in "B". Besides these, the alias2 rule includes the alias1 rule through <include-rules> tag. The alias 3 rule includes through <include-items> tag all the asset entries having their name starting with "C" and excludes through <exclude-items> tag all the asset entries having their names ending in "C". Besides these, the alias3 rule includes the alias2 rule through <include-rules> tag. In this scenario, the asset2 rule includes all the asset entries having their name starting with "A" or "B" and excludes all the entries with name ending in "A" or "B". On the other hand, the asset3 rule includes all the asset entries having their name starting with "B" or "C" and excludes all the entries with name ending in "B" or "C". Please note that the <include-rule> of the included rule is not considered in the container rule. Only the direct definition of items through <include-items> and <exclude-items> are taken into account. The <include-rules> tag appears no more than once under a rule tag and regards the definition of the rule alias patterns of the rules to be include in our rule. This tag is optional and has no attributes, only children <include-rule> tags.
`<include-rule>`	This tag appears at least once under an <include-rules> tag of a rule and regards the definition of one rule alias pattern that refer to rules that are included in our rule. This tag has one mandatory attribute. Attribute: • **alias** (mandatory): the rule alias pattern of the rules to be included in our rule, this attribute accepting wildcards.

`<exclude-rules>`	Besides defining its own excluded items, a rule can exclude the items filtered (included minus excluded) directly by other rules, directly meaning using <include-items> and <exclude-items> tags. These rules are referenced by their alias (an alias should be assigned to rules as needed). Excluding a rule defined for a different type by using the <exclude-rules> tag takes into consideration the asset entry name patterns defined through <include-items> and <exclude-items> tags and applies them to the items of the container rule type, the type of the included rule being ignored for flexibility. For example, let's consider 3 rules: alias1, alias2 and alias3. Let's say the alias1 rule includes through <include-items> tag all the asset entries having their name starting with "A" and excludes through <exclude-items> tag all the asset entries having their names ending in "A". The alias2 rule includes through <include-items> tag all the asset entries having their name starting with "A" or "B" and excludes through <exclude-items> tag all the asset entries having their names ending in "A" or "B". Besides these, the alias2 rule excludes the alias1 rule through <exclude-rules> tag. The alias 3 rule includes through <include-items> tag all the asset entries having their name starting with "A", "B" or "C" and excludes through <exclude-items> tag all the asset entries having their names ending in "A", "B" or "C". Besides these, the alias3 rule excludes the alias2 rule through <exclude-rules> tag. In this scenario, the asset2 rule includes all the asset entries having their name starting with "A" or "B" and those ending in "A" (from the excluded rule) and excludes all the entries with name ending in "A" or "B" and starting with A (from the excluded rule). Simplifying, the alias2 rule includes the entries with name starting with "B" and excludes those with name ending in "B". On the other hand, the asset3 rule includes all the asset entries having their name starting with "A" or "C" and excludes all the entries with name ending in "A" or "C". Please note that the <exclude-rule> of the excluded rule is not considered in the container rule. Only the direct definition of excludes through <exclude-items> and <include-items> are taken into account. The <exclude-rules> tag appears no more than once under a rule tag and regards the definition of the rule alias patterns of the rules that are excluded in our rule. This tag is optional and has no attributes, only children <exclude-rule> tags.
`<exclude-rule>`	This tag appears at least once under an <exclude-rules> tag of a rule and regards the definition of one rule alias pattern that refer to rules that are excluded by our rule. This tag has one mandatory attribute. Attribute: • **alias** (mandatory): the rule alias pattern of the rules that are excluded by our rule, this attribute accepting wildcards.

Simple Configuration File Example

Consider the next partitioning configuration:

```
<?xml version="1.0" encoding="utf-8"?>
<shaper-configuration xmlns="http://tempuri.org/ConfigSchema.xsd">
    <partitions directory=".">
        <partition name="Master" master="true">
            <compression>
                <lib type="Bitmap"/>
            </compression>
        </partition>
    </partitions>
</shaper-configuration>
```

To partition an asset, asset.bin, using the partitioning configuration, config.xml, run this command:

```
AssetTool -shape -p config.xml asset.bin
```

Using the configuration above, the partitioning tool will create one partition (asset) named Master in the current directory.This partition will contain all elements present in the original asset, the Bitmap elements being compressed.

Complex Configuration File Example

Consider the next complex partitioning configuration:

```
<?xml version="1.0" encoding="utf-8"?>
<shaper-configuration xmlns="http://tempuri.org/ConfigSchema.xsd">
    <partitions directory="C:/temp" block-size="3KB" reserve-space-names="20%" reserve-space-tre
        <compression>
            <lib type="Bitmap"/>
            <lib type="VertexBuffer"/>
        </compression>
        <partition name="Scenes" master="true" reserve-space-asset="256KB">
        </partition>
        <partition name="BitmapsAndFonts" reserve-space-lib="10%">
            <compression>
                <lib type="Font"/>
            </compression>
            <rule alias="RuleOne" type="Font">
                <include-items>
                    <include-item item="*one*"/>
                <include-item item="Anim*"/>
                </include-items>
                <exclude-items>
                    <exclude-item item="*two*"/>
```

```
                </exclude-items>
            </rule>
            <rule alias="RuleN" type="Font">
                <include-items>
                        <include-item item="*end"/>
                </include-items>
                <exclude-items>
                    <exclude-item item="Start*"/>
                </exclude-items>
            </rule>
            <rule type="Bitmap" item="*" reserve-space-lib="20%"/>
            <rule type="Bitmap" item="*BMP*" case-sensitive="true" exclude-from-partition="true"
        </partition>
        <partition name="VertexBuffers" reserve-space-tree="10%">
            <rule type="VertexBuffer" item="Vert*">
        <include-items>
                <include-item item="FirstVert*"/>
        </include-items>
        <include-rules>
            <include-rule alias="RuleOne"/>
            </include-rules>
            <exclude-rules>
                <exclude-rule alias="RuleN"/>
            </exclude-rules>
        </rule>
        </partition>
    </partitions>
</shaper-configuration>
```

Using the above configuration, the partitioning tool will create three partitions (assets) with the names: Scenes, BitmapsAndFonts and VertexBuffers, that will be found at C:/temp. The block size is 3KB, meaning that a data item will start only at the beginning of a new 3KB storage block.

The partition named Scenes is the master partition, containing any element of the original asset that is not contained in the other two partitions. In this partition, 20% of the name table space will be reserved at the end of the name table, 4KB will be reserved at the end of each asset lib look-up tree, and 256KB will be reserved at the end of the asset's data block. If there are VertexBuffer entries in this first partition, they'll be compressed (if there were Bitmap entries, they would be compressed two, but there is none since all go into the BitmapsAndFonts partition).

The partition named BitmapsAndFonts will contain all fonts of which names start with "Anim" or "Start" or contain "one" or end with "end" (wildcards is used) excluding all the entries with names that contain "two". Also, this partition includes all the bitmaps of the original asset, except for those that contain "BMP" in their name, case-sensitive, which are excluded from this partition and, without other rules to target them, are included in the master partition. In this partition, 20% of the name table space will be reserved at the end of the name table, 4KB will be reserved at the end of each asset lib look-up tree, and 10% of the total lib data block will be reserved at the end of each lib's data block, except for the Bitmaps lib for which 20% will be reserved. Compression is applied to all Bitmap elements (because of the global compression rule involving this type of elements) and to all Font elements (because of the local partition compression rule involving those elements).

Finally the last partition, VertexBuffers, will contain all the VertexBuffer entries of the original asset that have names that start with "Vert" (direct item definition - equivalent to defining an <include-item>) or "FirstVert" or contain "one" or start with "Anim" except all the entries with names that contain "two" (see <include-rule> definition). Moreover, this partition will contain all the VertexBuffer entries with names that start with "Start", being excluded from the partition all those with names that end in "end" (see <exclude-rule> definition). In this partition, 20% of the name table space will be reserved at the end of the name table and 10% of the look-up tree space will be reserved at the end of each asset lib look-up tree. Compression is applied to all VertexBuffer elements (see local compression rule for this third partition).

In conclusion, in what concerns the partitioning rules, the more specific definitions override the more general ones. Regarding the compression rules, if there are compression rules defined globally (under partitions tag), they are inherited to all the assets outputted by the partitioning process.

Revision #6
Created 2023-02-16 01:21:09 UTC by Tsuyoshi.Kato
Updated 2025-01-24 10:41:03 UTC by Elisabeth Zauner