Renderer module

The renderer module executed renderer commands in the censhare Web UI.

Renderer commands

This section describes some of the most important renderer commands. Besides a short description what each command does, the method name and all special parameters are given. Parameters are normally defined as attributes of the `command` node. All commands have the following two standard parameters.

- *document_ref_id*: The ID of the document to be processed.
- *method*: The name to define the specific command.

Additionally, a short XML example is given of how to define each command.

Change Page Size Command

This command can be used to change the size of all pages of a document. Therefore, one page width and one page height can be specified.

Method Name

- *change-page-size*

Parameters

- *page_width*: New width of the document pages in points.
- *page_height*: New height of the document pages in points.

Example

```
<command document_ref_id="1" method="change-page-size" page_width="300" page_height="500"/>
```

Check Missing Fonts Command

This command can be used to check if a document contains any missing fonts. If so, an exception is thrown.

Method Name

- *check-missing-fonts*

Parameters

No special parameters are needed.

Example

```
<command document_ref_id="1" method="check-missing-fonts"/>
```

Check Missing Links Command

This command can be used to check if a document contains any missing image links. It aborts the current command sequence in case that any image link in the document is marked as missing. This command is typically included when creating a PDF. Note, that the command ignores missing text links, since texts are embedded into the layout anyway and hence missing or modified links do not lead to low quality print results as is the case for images.

Method Name

- *check-missing-links*

Parameters

No special parameters are needed.

Example

```
<command document_ref_id="1" method="check-missing-links"/>
```

Check Modified Links Command

This command can be used to check if a document contains any modified image links. It aborts the current command sequence in case that any image link in the document is marked as modified, i.e. a content update is outstanding. This command is typically included when creating a PDF. Note, that the command ignores modified text links, since texts are embedded into the layout anyway and hence missing or modified links do not lead to low quality print results as is the case for images.

Method Name

- *check-modified-links*

Parameters

No special parameters are needed.

Example

```
<command document_ref_id="1" method="check-modified-links"/>
```

Check Placed Collections Command

This command can be used to check if all placements of a document, which are placed via a collection (layout group transformation), are still valid. If any significant attributes, e.g. the language, have changed, the collections are placed again. This is effectively the same as executing all layout group transformations again, but more efficient, since placements are only changed if necessary.

This command does not only check if the source asset of a placement has changed, but also if URL parameters of placements done via transformations are different (e.g. another target-asset-id or group-name). If so, the transformation must be re-executed. This command is typically used after the asset structure of a layout was duplicated.

Method Name

- *check-placed-collections*

Parameters

- *picture_placement_rule*: Specifies how to position a picture when placed into a box. Possible values are:
    - `centered`,
        - `centered-proportional` (this is used as default, if the parameter is not specified),
        - `fit`,
        - `fit-proportional`,
        - `fit-box`,
        - `top-left`.

Example

```
<command document_ref_id="1" method="check-placed-collections" picture_placement_rule="fit-proportional"/>
```

Close Command

This command can be used to close an previously opened document.

Method Name

- *close*

Parameters

No special parameters are needed.

Example

```
<command document_ref_id="1" method="close"/>
```

Copy Pagearea Asset Command

This command can be used to copy the content of a pagearea asset on a page or into a target box (i.e. other pagearea) at given position in points (relative to parent component). Note: The pagearea asset is not actually placed (i.e. no relation is created), just it's content is copied.

Method Name

- *copy-pagearea-asset*

Parameters

- *uid*: The temporary unique ID of the pagearea box.
- *parent_uid*: The unique ID of the parent component.
- *xoffset*: The x-offset in points relative to the parent component, where 0 is the default if not defined.
- *yoffset*: The y-offset in points relative to the parent component, where 0 is the default if not defined.
- *width*: The new width in points, where 0 is the default if not defined.
- *height*: The new height in points, where 0 is the default if not defined.
- *place_asset_id*: The ID of the pagearea asset, which should be placed.
- *place_asset_element_id*: The index of the asset element of the pagearea asset which should be placed.
- *uid-mapping*: Optional: Defined as zero, one or more child nodes to map an UID (normally of a box) to a new temporary UID. This mapping is then used in the following render commands as well. The node with name *uid-mapping* contains two attributes:
    - *from*: The old unique ID.
    - *to*: The new unique ID.

Example

```
<command document_ref_id="1" method="copy-pagearea-asset" uid="#1002" parent_uid="p1" place_asset_element_id="1"
        place_asset_id="502770" xoffset="11.943127" yoffset="11.943127">
  <uid-mapping from="b282" to="#1005"/>
  <uid-mapping from="b281" to="#1004"/>
</command>
```

Copy Pagearea Command

This command can be used to copy the content of a pagearea box from one document to another.

Method Name

- *copy-pagearea*

Parameters

- *uid*: The temporary unique ID of the pagearea box.
- *parent_uid*: The unique ID of the parent component.
- *xoffset*: The x-offset in points relative to the parent component, where 0 is the default if not defined.
- *yoffset*: The y-offset in points relative to the parent component, where 0 is the default if not defined.
- *width*: The new width in points, where 0 is the default if not defined.
- *height*: The new height in points, where 0 is the default if not defined.
- *source_document_ref_id*: The ID of the source document from where the pagearea should be copied.
- *source_uid*: The unique ID of the pagearea box from the source document.
- *uid-mapping*: Optional: Defined as zero, one or more child nodes to map an UID (normally of a box) to a new temporary UID. This mapping is then used in the following render commands as well. The node with name *uid-mapping* contains two attributes:
    - *from*: The old unique ID.
    - *to*: The new unique ID.

Example

```
<command method="copy-pagearea" source_document_ref_id="1" source_uid="b100" document_ref_id="2" uid="#1002"
        parent_uid="p1" xoffset="12.6" yoffset="11.3">
  <uid-mapping from="b282" to="#1005"/>
  <uid-mapping from="b281" to="#1004"/>
</command>
```

Correct Document Command

This command can be used to validate a document and correct it's slugs and child element relations, if necessary. Typically it is used for documents that were created by duplication or when applying a copy template.

Method Name

- *correct-document*

Parameters

- *force-correction*: If `true` the document is automatically corrected, if necessary, otherwise an exception is thrown. The default, if not defined, is `false`.
- *force-conversion*: If `true` an existing/old layout document is always converted to use the virtual file system. The default, if not defined, is `false`.
- *inline-vfs-text-placements*: If this flag is given, it overrides the InDesign preference setting (attribute "inline-vfs-text-placements" in node "indesign") to (not) inline VFS text placements during correction of the document.

Example

```
<command document_ref_id="1" method="correct-document" force-correction="true", force-conversion="true"/>
```

Debug Command

This command can be used to to debug the current request of the renderer and the currently open documents. If such a command is contained in the command sequence the document will not be opened in hidden mode - it will be opened visible. When the debug step is reached a confirmation dialog is shown and the execution will stop until the user has pressed either "ok" or "cancel".

Note that this command will be ignored if the renderer runs in headless mode.

Method Name

- *debug*

Parameters

- *report*: If `true` a document report is requested and written into XML logging window. The default, if not defined, is `true`.
- *show-dialog*: If `true` a confirmation dialog is shown and the execution will stop until the user has pressed either "ok" or "cancel". The default, if not defined, is `true`.

Example

```
<command document_ref_id="1" method="debug"/>
```

Delete Boxes Command

This command can be used to to delete boxes from the document. Boxes can be specified either directly via a box element with a box uid or indirectly via a group element and a group id. All corresponding boxes (together with potentially child boxes) will then be deleted.

Note

This command already takes care of updating the asset element structure (and synchronizing (element) relations with placements) after deleting boxes with placements.

Method Name

- *delete-boxes*

Parameters

- *box*: Node name of zero, one or more child nodes defining a box to be deleted.
  - *uid*: Attribute of the box node defining the unique ID of the box.
- *group*: Node name of zero, one or more child nodes defining a group of which each box is to be deleted.
  - *id*: Attribute of the group node defining the ID of the layout group.

Example

```
<command method="delete-boxes" document_ref_id="1">
  <box uid="b220"/>
  <group id="1"/>
  <box uid="b221"/>
  <group id="2"/>
</command>
```

Detach Command

This command can be used to detach a placement from a specific box. The box itself will not be deleted.

Method Name

- *detach*

Parameters

- *uid*: The unique ID of the box where to detach from.

Example

```
<command document_ref_id="1" method="detach" uid="b220"/>
```

Detach Pagearea Command

This command can be used to detach a pagearea box and all of it's subcomponents from a document.

Method Name

- *detach-pagearea*

Parameters

- *uid*: The unique ID of the pagearea box where to detach from.

Example

```
<command document_ref_id="1" method="detach-pagearea" uid="b220"/>
```

Dissolve Command

This command can be used to dissolve the file link of a placed text (with or without a transformation). That text is than embedded into the document and the corresponding box is unlocked and can be freely modified regarding content and style.

Note that the embedded text looses its relation to the corresponding censhare text asset. So it's no longer possible to run a content update on the affected text box.

Method Name

- *dissolve*

Parameters

- *uid*: The unique ID of the text box that should be dissolved.

Example

```
<command document_ref_id="1" method="dissolve" uid="b220"/>
```

Edit Placement Command

This command can be used to change the scale, offset and rotation of a placement in a picture box.

Method Name

- *edit-placement*

Parameters

- *uid*: The unique ID of the picture box to be edited. If this refers to a non-picture box an exception is thrown.
- *xscale*: A double value for scaling in x-direction.
- *yscale*: A double value for scaling in y-direction.
- *rotation*: A doulbe value in angular degree for rotation.
- *xoffset*: A double value for the x-offset in points.
- *yoffset*: A double value for the y-offset in points.

Example

```
<command method="edit-placement" document_ref_id="1" asset_element_id="24"
      rotation="90" xscale="0.2" yscale="0.2" xoffsetmm="50" yoffsetmm="20"/>
```

Exchange Opi Command

This command can be used to replace all high resolution picture against OPI pictures or vice versa. That is all picture placements corresponding to a *master* storage of an asset are re-linked to the corresponding *opi* storage item (or vice versa).

Method Name

- *exchange-opi*

Parameters

- *place_storage_key*: If option `opi` is used all high resolution pictures are replaced against OPI pictures. For any other option (e.g. `master`) all OPI pictures are replaced against high resolution pictures.

Example

```
<command method="exchange-opi" document_ref_id="1" place_storage_key="opi"/>
```

Export Command

This command can be used to create a PDF, EPS, IDML or a preview file from a document. This file is then copied to the server and the corresponding file system and file path are returned.

Method Name

- *pdf*, *eps*, *idml* or *preview*

Parameters

- *format*: In case of method name *preview* this defines the preview format of the files to be created. Possible values are `JPEG`, `TIFF` and `GIFf`. If missing the default `JPEG` is used.
- *asset_element_id*: Index of the asset element for which the corresponding layout element (normally page or box) should be used to create the PDF/EPS/IDML/preview file from. If this is optional parameter is omitted, the corresponding files for the whole document are created.
- *spread*: In case of method name *pdf* this boolean flag decides if a PDF file for each spread should be created. If missing the default `false` is used.
- *pdf_stylename*: In case of method name *pdf* this is the specific PDF style (e.g. [Press Quality]) as defined in InDesign. If missing the empty string `''` is used as default.
- *scale*: In case of method name *eps* or *preview* this is used to define the scaling factor. If missing the default `1.0` is used.
- *preview*: In case of method name *eps* this defines the EPS preview format. Possible values are `pict`, `tiff` and `none`. If missing the default `pict` is used.
- *jpeg_quality*: In case of method name *preview* this defines the JPEG quality. Possible values are `low`, `good`, `excellent` and `great`. If missing the value defined in the preferences (`<create-preview jpeg-quality="good" ...>`) is used. If this is missing as well the default `excellen` is used.

Example

Input:

```
<command method="preview" format="JPEG" scale="0.5" document_ref_id="1"/>
```

Output:

```
<command method="preview" format="JPEG" scale="0.5" document_ref_id="1">
  <file element_idx="1" corpus:asset-temp-filepath="file:335008.jpg" corpus:asset-temp-filesystem="assets-temp"
/>
  <file element_idx="2" corpus:asset-temp-filepath="file:335009.jpg" corpus:asset-temp-filesystem="assets-temp"
/>
</command>
```

Export Document Command

This command can be used to export a given document. The placed pictures/texts of the document are exported and the placements are exchanged (asset file paths are replaced by local paths), all slugs are removed from the document and the document file is exported itself. The whole export folder is then zipped and the ZIP file is copied to the server (file references are reported).

Method Name

- *export-document*

Parameters

- *document_ref_id*: ID of the document to be exported as specified in the open command.
- *export-placed-media*: If `true` the placed pictures are exported and the placements are exchanged to the exported files. The default is `false`.
- *export-placed-texts*: If `true` the placed (InCopy) texts are exported and the placements are exchanged to the exported files. The default is `false`.
- *create-links-folder*: If `true` the pictures and/or texts are exported into a "Links" subfolder. The default is `false`.

Example

Input:

```
<command document_ref_id="1" method="export-document" export-placed-media="true" export-placed-texts="true"
create-links-folder="true"/>
```

Result:

```
<command document_ref_id="1" method="export-document" export-placed-media="true" export-placed-texts="true"
create-links-folder="true">
  <file corpus:asset-temp-filepath="file:275483.zip" corpus:asset-temp-filesystem="temp"/>
</command>
```

Mark Not Printable Boxes Command

This command can be used to find boses that will be marked as "not printable". This command is typically executed before an output format (PDF, EPS) of the document is created. The target boxes, which must be marked, are identified by XPath filters, which filter either placed assets or boxes.

Method Name

- *mark-not-printable-boxes*

Parameters

- *box-filter*: A child node with this names defines the XPath filter for getting the boxes to be marked.
- *placement-filter*: A child node with this names defines the XPath filter for getting the placements of which the corresponding boxes are to be marked. This or *box-filter* or both can be used. If none of these parameters are given, the command does nothing.

Example

```
<command document_ref_id="1" method="mark-not-printable-boxes">
  <box-filter condition=":element/@content='text' and element/@page='p3'"/>
</command>
```

Move Box Command

This command can be used to move a pagearea or box and all of it's subcomponents within a document, optionally resizing the component.

Method Name

- *move-box* or *move-pagearea*

Parameters

- *uid*: The unique ID of the component to be moved.
- *parent_uid*: The unique ID of the parent component in which the movement should be done.
- *xoffset*: The x-offset in points relative to the parent component, where 0 is the default if not defined.
- *yoffset*: The y-offset in points relative to the parent component, where 0 is the default if not defined.
- *width*: The new width in points, where 0 is the default if not defined.
- *height*: The new height in points, where 0 is the default if not defined.

Example

```
<command document_ref_id="1" method="move-box" uid="b100" parent_uid="p1" xoffset="11.943127"
        yoffset="11.943127" width="90" height="100"/>
```

Open Command

This command can be used to open a document and is the beginning of every render command sequence.

Method Name

- *open*

Parameters

- *asset_id*: The asset ID of the asset of which the document should be opened.
- *asset_ref_id*: The asset ref ID of the asset of which the document should be opened.
- *report_mode*: This option specifies the detail level of the document report when opening. Possible values are `minimal`, `geometry` and `maximal`. If missing the default `geometry` is used.
- *structure_mode*: This option specifies the detail level of the XML structure included in the document report when opening. Possible values are `min imal`, `structure` and `content`. If missing the default `structure` is used.
- *inline-vfs-text-placements*: If this flag is given, it overrides the InDesign preference setting (attribute "inline-vfs-text-placements" in node "indesign") to (not) inline VFS text placements during opening of the document.

Example

```
<command method="open" asset_id="241110" document_ref_id="1"/>
```

Place Collection Command

This command can be used to place a collection asset onto a document. Either into a dedicated target group or into any group.

Method Name

- *place-collection*

Parameters

- *group_id*: The ID of the group to place the collection asset into. If missing the parameter *group_name* is used to get the "placement target".
- *group_id*: The name of the group to place the collection asset into. If missing the parameter *uid* is used to get the "placement target".
- *uid*: The unique ID of the box to place the collection asset into. If missing the first (any) group (normally there should be just one in that case) is used as "placement target".
- *place_asset_id*: The ID of the collection asset to be placed.
- *transformation_key*: The key of the transformation asset to transform the collection asset before placing.
- *picture_placement_rule*: Specifies how to position a picture when placed into a box. Possible values are:
    - `centered`,
    - `centered-proportional` (this is used as default, if the parameter is not specified),
    - `fit`,
    - `fit-proportional`,
    - `fit-box`,
    - `top-left`.

Example

```
<command document_ref_id="1" method="place-collection" group_id="3" place_asset_id="241110"
        transformation_key="censhare:product-transformation" picture_placement_rule="fit-proportional"/>
```

Place Command

This command can be used to place an asset into a specific box.

Method Name

- *place*

Parameters

- *uid*: The unique ID of the box where to place the given asset into.
- *place_asset_id*: The ID of the asset to be placed.
- *place_storage_key*: The key of the storage item of the asset to be placed. If not defined and if no transformation node is given the meta data of the asset are placed.
- *place_asset_element_id*: The index of the asset element to get the storage item from. This is only used if *place_storage_key* is given.
- *transformation*: Node name of the optional child node to define a transformation (containing attributes `key`, `url` and `format`).
- *ignore_content_update*: If `false` and if an older version of the asset is already placed, an exception is thrown. The default, if not defined, is `false` .
- *child_asset_rel*: Node name of the optional child node to define a template of the child relation to be created.
- *child_asset_element_rel*: Node name of the optional child node to define a template of the child element relation to be created.
- *is_manual_placement*: If this optional flag is given, the corresponding flag is set on the child element relation to be created.
- *keep_box_attributes*: If `true` an existing content on the box should not be deleted. The default, if not defined, is `false`.
- *group_transformation_url*: With this optional parameter a group transformation URL can be defined.
- *update-content-elements*: If `true` for an unlocked text box only the content of censhare content tags is updated, but style changes are kept as far as possible. The default, if not defined, is `false`.

Example

```
<command method="place" document_ref_id="1" uid="b220" asset_element_id="13" place_asset_id="452370"
        place_asset_element_id="0" place_storage_key="master"/>
```

Place Snippet Command

This command can be used to place an InDesign snippet asset onto a page. Optionally, a content asset (e.g. a product) is additionally placed into every group of the placed snippet.

Method Name

- *place-snippet*

Parameters

- *parent_uid*: The unique ID of the page to place the snippet onto.
- *snippet_asset_id*: The ID of the snippet asset to be placed.
- *snippet_asset_element_id*: The index of the asset element of the snippet asset which is to be placed. If not defined the actual root element is used.
- *xoffset*: The x-offset in points relative to the parent page, where 0 is the default if not defined.
- *yoffset*: The y-offset in points relative to the parent page, where 0 is the default if not defined.
- *place_asset_id*: Optional: The ID of the content asset (e.g. a product) to be placed into every group of the placed snippet.

Example

```
<command document_ref_id="1" method="place-snippet" parent_uid="p1" snippet_asset_element_id="0"
snippet_asset_id="502770"
        xoffset="11.943127" yoffset="11.943127" place_asset_id="502771"/>
```

Rename Group Command

This command can be used to change the name of a layout group.

Method Name

- *rename-group*

Parameters

- *group_id*: The ID of the group to change the name.
- *group_name*: The new name of the layout group.

Example

```
<command document_ref_id="1" method="rename-group" group_id="3" group_name="Group C"/>
```

Report Command

This command can be used to request a document report.

Method Name

- *report*

Parameters

- *expand*: This option specifies the detail level of the document report. Possible values are `minimal`, `geometry` and `maximal`. If missing the default `geometry` is used.
- *structure*: This option specifies the detail level of the XML structure included in the document report. Possible values are `minimal`, `structure` and `content`. If missing the default `structure` is used.

Example

```
<command document_ref_id="1" method="report" expand="minimal" structure="content"/>
```

Save Command

This command can be used to save a document into a file. This file is then copied to the server and the corresponding file system and file path are returned. Additionally, the application version of the document is written into the XML.

Method Name

- *save*

Parameters

No special parameters are needed.

Example

Input:

```
<command method="save" document_ref_id="1"/>
```

Output:

```
<command method="save" document_ref_id="1" corpus:asset-temp-filepath="file:335010.indd"
        corpus:asset-temp-filesystem="assets-temp" corpus:app_version="8.1"/>
```

Script Command

This command can be used to execute an InDesign script (JavaScript, AppleScript or VBScript). It is possible to define two different script output results (using function `app.scriptArgs.setValue`). With name "censhareResult" a general script output value can be defined which is then written into the `result` attribute of the `script` child node. With name "censhareResultFiles" an escaped XML string can be defined which is used to reference files and folders created during the script. The syntax of this files XML is like this:

```
<files>
  <file key="key1" path="/Users/xx/Desktop/fileFromScript.txt"/>
  <file key="key2" path="/Users/xx/Desktop/folderFromScript"/>
</files>
```

These files are then copied to the server and the corresponding file system and file path are returned in a `files` child nodes of the `script` node (see examples for details).

Method Name

- *script*

Parameters

- *script_asset_id*: The ID of the script asset (resource asset) to get the scripts (there may be multiple script master storage items) from.
- *script*: If the above parameter is not defined, a child node with this name can be used to define an "inline" script. With the attribute `language` of this node the language of the script can be defined.
- *param*: Zero, one or more script input parameter can be defined using child nodes with this name (having attributes `name` and `value`.
- *boxes*: A child node with this name can be used to define boxes as "censhareTargetBoxes" as script input parameter. This node can contain one or more child nodes with name `box` which has an attribute `uid`.

Examples

Input:

```
<command document_ref_id="1" method="script" script_asset_id="35673">
  <param name="input1" value="1"/>
  <param name="input2" value="2"/>
  <boxes>
    <box uid="b220"/>
  </boxes>
</command>

<command method="script" document_ref_id="1">
  <script language="javascript">
    app.scriptArgs.setValue("censhareResult", "This is the script result");
  </script>
</command>

<command method="script" document_ref_id="1">
  <script language="javascript">
    /* use script to create a file /Users/xx/Desktop/file1FromScript.txt and a folder /Users/xx/Desktop
/folderFromScript */
    var resultFilesString = "&lt;files&gt;&lt;file key=\"key1\" path=\"/Users/xx/Desktop/fileFromScript.txt\"
/&gt;&lt;file key=\"key2\" path=\"/Users/xx/Desktop/folderFromScript\"/&gt;&lt;/files&gt;";
    app.scriptArgs.setValue("censhareResultFiles", resultFilesString);
  </script>
</command>
```

Output:

```
<command document_ref_id="1" method="script" script_asset_id="35673">
  <param name="input1" value="1"/>
  <param name="input2" value="2"/>
  <boxes>
    <box uid="b220"/>
  </boxes>
</command>

<command method="script" document_ref_id="1">
  <script language="javascript" result="This is the script result">
    app.scriptArgs.setValue("censhareResult", "This is the script result");
  </script>
</command>

<command method="script" document_ref_id="1">
  <script language="javascript">
    /* use script to create a file /Users/xx/Desktop/file1FromScript.txt and a folder /Users/xx/Desktop
/folderFromScript */
    var resultFilesString = "&lt;files&gt;&lt;file key=\"key1\" path=\"/Users/xx/Desktop/fileFromScript.txt\"
/&gt;&lt;file key=\"key2\" path=\"/Users/xx/Desktop/folderFromScript\"/&gt;&lt;/files&gt;";
    app.scriptArgs.setValue("censhareResultFiles", resultFilesString);
    <files>
      <file key="key1" path="/Users/xx/Desktop/fileFromScript.txt" corpus:asset-temp-filepath="file:335024.txt"
corpus:asset-temp-filesystem="assets-temp"/>
      <file key="key2" path="/Users/xx/Desktop/folderFromScript" corpus:asset-temp-filesystem="assets-temp"
folder="true" corpus:asset-temp-filepath="file:335025.zip"/>
    </files>
  </script>
</command>
```

Text Content Command

This command can be used to export the text content of a document into into different files for one or every page. These files are then copied to the server and the corresponding file systems and file paths are returned.

Method Name

- *text-content*

Parameters

- *mimetype*: The mime type of the text content files. If null the text preview mime type as defined in the InDesign preferences is taken.
- *asset_element_id*: The index of the document asset element of which the corresponding page should be used to export the text content. If not given the text content of all pages is export into an own file.

Example

Input:

```
<command document_ref_id="1" method="text-content" mimetype="application/vnd.adobe.indesign-idms"/>
```

Output:

```
<command document_ref_id="1" method="text-content" mimetype="application/vnd.adobe.indesign-idms">
  <file element_idx="1" corpus:asset-temp-filesystem="assets-temp" corpus:asset-temp-filepath="file:335024.
idms" mimetype="application/vnd.adobe.indesign-idms"/>
  <file element_idx="2" corpus:asset-temp-filesystem="assets-temp" corpus:asset-temp-filepath="file:335025.
idms" mimetype="application/vnd.adobe.indesign-idms"/>
</command>
```

Update Asset Element Structure Command

This command can be used to update the asset elements (actual and optionally target) of the document asset to be in sync with the document structure.

Method Name

- *update-asset-element-structure*

Parameters

- *force*: If `false` and if not one of the previous render commands requires an updating of the asset element structure nothing is done here. The default is `false`.
- *sync-target-elements*: If `true` target asset elements are synchronized from actual asset elements. The default is `false`.
- *delete-target-elements*: If `true` and if *sync-target-elements* is `false` all existing target elements with hierarchy level greater than one are deleted. The default is `false`. If both *sync-target-elements* and *delete-target-elements* are `false` all target elements are kept as they are.

Example

```
<command document_ref_id="1" method="update-asset-element-structure" force="true" sync-target-elements="true"/>
```

Update Auto Client Variant Command

This command can be used to update all layout geometry variants of the document asset. Therefore, the current document is closed, the original document is copied from server and opened. All elements (actual and target) from document are replaced with elements from the original document. All placed texts of current document, where any version of the text is placed on the master document too, are imported again with the current version of the text in the variant.

Method Name

- *update-auto-client-variant*

Parameters

No special parameters are needed.

Example

```
<command document_ref_id="1" method="update-auto-client-variant"/>
```

Update Content Command

This command can be used to update the content of all placements of either one or all placed assets.

Method Name

- *update-content*

Parameters

- *asset_id*: The ID of the placed child asset of which the placements should be updated. If not given all placed assets are considered for the content update.
- *ignore-if-size-changed*: If `true` and if *asset_id* is not given a placed asset where only the size has changed is ignored during for the content update. The default is `false`.
- *sizeChangeToleranceMm*: The amount of millimeters of when an asset is said to be changed in size. The default is `0.0`.
- *force*: This flag is deprecated and does not influence the content update at all.
- *updateContentElements*: If `true` unlocked text boxes are updated by synchronizing the censhare content tags and keeping but style changes as far as possible. If `false` such boxes are ignored. If this parameter is not given the attribute `update-content-elements-in-server-modeupdate-content-elements-in-server-mode` from the InDesign preferences is taken where `false` is the default.
- *target*: Zero, one or more box parameters can be defined by using child nodes with this name (having attribute `uid` and transformation child node).

Example

```
<command method="update-content" document_ref_id="1" asset_id="241120"/>
```

Update Geometry Command

This command can be used to execute a geometry of a document layout asset.

Method Name

- *update-geometry*

Parameters

- *delete-target-rels*: If `true` (which is the default) after the geometry update all target element relations are deleted where the following conditions are met:
    - No geometry update flag is set on the relation.
    - A corresponding actual element relation exists.
    - The source_sid attribute of the actual element relation points to this target relation.

Example

```
<command method="update-geometry" document_ref_id="1"/>
```

Update Meta Data Command

This command can be used to update the meta data (layout box tags) of a given tagged text box. Therefore, it is searched for elements of the form.

```
<censhare:content configurationAssetKey="id">
```

in the XML structure of the box's content.

Method Name

- *update-meta-data*

Parameters

- *uid*: The unique ID of the box to update its meta data.
- *meta-data*: Node name of the child node to define the update values of the layout box tags. The `meta-data` node can contain one or more child nodes with name `item` having an attribute `id` and a text child for the new value (see example).

Example

```
<command method="update-meta-data" document_ref_id="1" uid="b220">
  <meta-data>
    <item id="censhare:layout-box-tag.name>Product A</item>
    <item id="censhare:layout-box-tag.price>12.99</item>
  </meta-data>
</command>
```

Update Placeholder Command

This command can be used to update XML tags of a given document from the attributes of the corresponding asset.

Method Name

- *update-placeholder*

Parameters

No special parameters are needed.

Example

```
<command method="update-placeholder" document_ref_id="1"/>
```