# XSLT Extensions

5th November 2013

censhare

For customers or partners with NDA

# Overview

censhare

- Definitions of Wikipedia
  - XSLT
    - XSLT (E**x**tensible **S**tylesheet **L**anguage **T**ransformation) is a declarative, XML-based language used for the transformation of XML documents.

# Overview

censhare

- censhare
  - censhare has own implementation of XML, XPath 2.0 and XSLT 2.0
    - Advantages
      - XML implementation is faster and has less objects in memory
      - XPath and XSLT implementation gives expandability without 3rd party license costs (like Saxon) and is faster
      - Additional XPath functions, e.g. asset query, additional xsl instructions (e.g. xml update) will follow
    - Disadvantages
      - Not the entire range is implemented
      - List of supported operators and functions is available
    - Documented functionality
      - This training shows only the supported functionality of the censhare implementation of version 4.4

# Overview

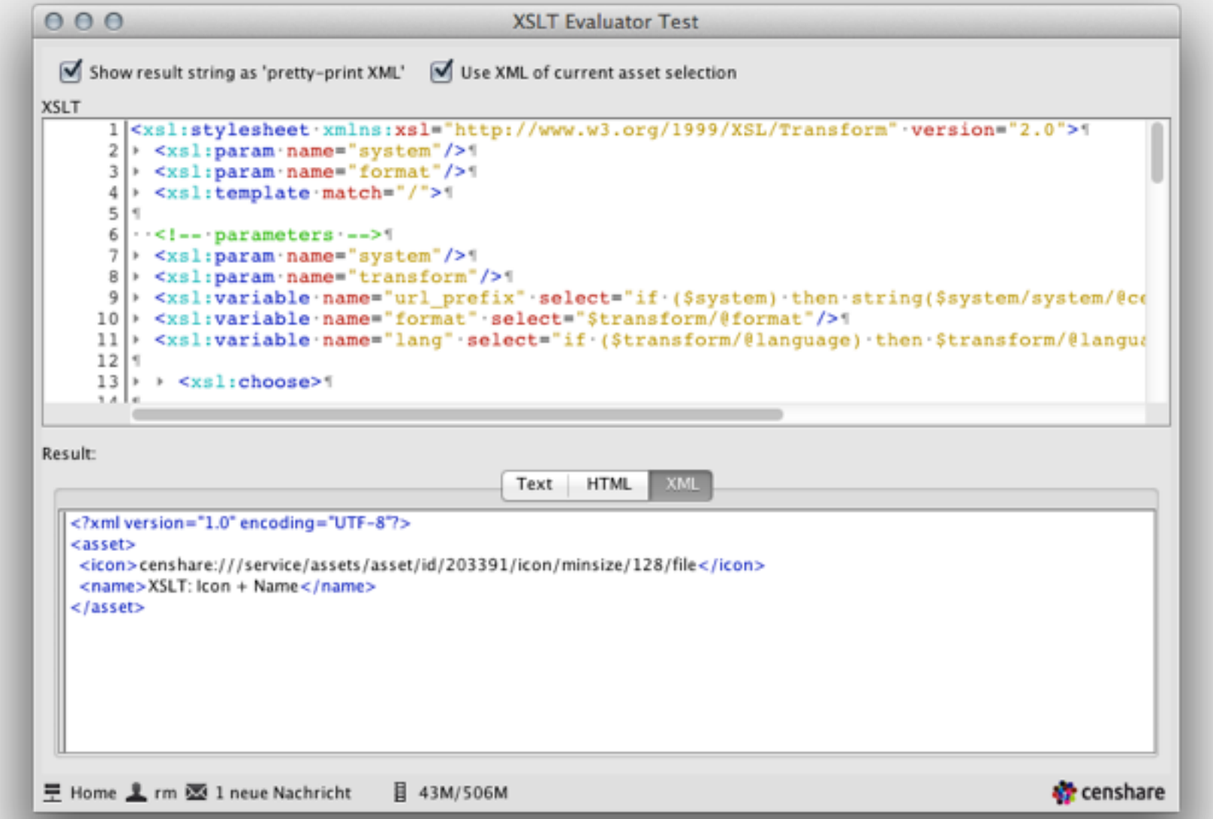
censhare

- censhare
  - XML is used in
    - Assets
    - Master data
    - Configurations, dialog definitons
  - XPath is used in
    - All dialog definitions
    - As filter in resource assets
    - As expression for custom censhare database indexes
    - All XSLT's
  - XSLT is used in
    - Asset info (about 350 XPath expressions in default customizing)
    - All transformations

# Overview

censhare

- censhare
  - Provides a test window in all censhare Clients in admin mode with menu item “XSLT Evaluator Test”
  - Current asset selection or own XML can be used as source



# censhare XSLT extensions

censhare

- Instructions
  - We have implemented own instructions to enhance XSLT
  - All censhare functions have the namespace “cs” for “**c**en**s**hare”. This namespace is not optional and must exist

# censhare XSLT extensions

censhare

- Instructions
  - cs:diff-items, cs:matching-items, cs:deleted-items, cs:inserted-items
    - Gives the option to compare given old and new string sequences and output matching, deleted and inserted items
    - “cs:matching-items”, “cs:deleted-items” and “cs:inserted-items” are optional
    - Attributes
      - “select-old”: Input of old string sequence items
      - “select-new”: Input of new string sequence items
      - “equals-function”: Optional reference to a equals function. If not exists, then exact match is used
    - Example

```
<cs:diff-items select-old="$a" select-new="$b" equals-function="my:equals"/>
<xsl:function name="my:equals" as="xs:boolean">
  <xsl:param name="a"/>
  <xsl:param name="b"/>
  <xsl:value-of select="upper-case($a) eq upper-case($b)"/>
</xsl:function>
```

# censhare XSLT extensions

censhare

- Instructions
  - cs:diff-items, cs:matching-items, cs:deleted-items, cs:inserted-items
    - Example

```
<xsl:variable name="old" select="tokenize('This is the old sentence.', ' ')"/>
<xsl:variable name="new" select="tokenize('This is the new version of the sentence', ' ')"/>
<xsl:template match="/">
  <cs:diff-items select-old="$old" select-new="$new">
   <cs:matching-items><xsl:value-of select="concat(string-join(., ' '), ' ')"/></cs:matching-items>
   <cs:deleted-items>
     <span style="background-color:red"><xsl:value-of select="concat(string-join(., ' '), ' ')"/></span>
   </cs:deleted-items>
   <cs:inserted-items>
     <span style="background-color:green"><xsl:value-of select="concat(string-join(., ' '), ' ')"/></span>
   </cs:inserted-items>
  </cs:diff-items>
</xsl:template>
```

# censhare XSLT extensions

censhare

- Instructions
  - cs:output
    - Defines result file of the transformation
    - Attributes
      - “href”: URL to file
      - “media-type”: MIME type of the result file
    - Example

```
<cs:output href=”asset/storage_item[@key='master']/@url” media-type=”asset/storage_item[@key='master']/@mimetype”/>
```

    - Returns
      - Master file of the selected asset as result of the transformation

# XSLT commands

censhare

- Instructions
  - General
    - XSLT commands are an extension to XSLT to access the censhare API
      - Asset management
      - Transformations (XSL, barcode, image, FOP)
      - Office
      - Context
      - I/O
      - Renderer
      - PDF
      - Social networks (Facebook, Twitter, Youtube)
      - E-mail

# XSLT commands

censhare

- Instructions
  - Structure

```
<cs:command name="[name of the command]" returning="[result variable]">
  <cs:param name="[name of the parameter]" select="[value of the parameter]"/>
  <cs:param name="[name of the parameter]" select="[value of the parameter]"/>
</cs:command>
```

# XSLT commands

censhare

- Instructions
  - XSLT commands
    - Results can be collected in 3 ways
      - Using the “returning” attribute (define variable with given name and store result in variable)

```
<cs:command name="com.censhare.api.assetmanagement.CloneAndCleanAssetXml" returning="resultAssetXml">
  <cs:param name="source" select="asset"/>
</cs:command>
<xsl:copy-of select="$resultAssetXml"/>
```

      - Using the result of the “cs:command” function

```
<cs:command name="com.censhare.api.assetmanagement.CloneAndCleanAssetXml">
  <cs:param name="source" select="asset"/>
</cs:command>
```

      - Using the “dest” parameter (not available at every command)

```
<cs:command name="com.censhare.api.io.WriteXML">
  <cs:param name="source" select="$metaDoc"/>
  <cs:param name="dest" select="concat($out, 'EPUB-Debug.xml')"/>
</cs:command>
```

# XSLT commands

censhare

- Asset management
  - Commands
    - Check in new asset
    - Check out asset
    - Check in asset
    - Check out abort asset
    - Update asset
    - Clone and clean asset XML
    - Delete asset

# XSLT commands

censhare

- Asset management
  - Check in new asset
    - Create asset of given asset XML
    - Attributes
      - “name”: “ com.censhare.api.assetmanagement.CheckInNew”
      - “returning”: XML of created asset
    - Parameters
      - “source”: asset XML to create

# XSLT commands

censhare

- Asset management
  - Check in new asset
    - Examples

```
<xsl:variable name="assetXml">
  <asset name="First test asset" type="temp."/>
</xsl:variable>
<xsl:variable name="resultAssetXml"/>
<cs:command name="com.censhare.api.assetmanagement.CheckInNew" returning="resultAssetXml">
  <cs:param name="source" select="$assetXml"/>
</cs:command>
```

```
<xsl:variable name="resultAssetXml"/>
<cs:command name="com.censhare.api.assetmanagement.CheckInNew" returning="resultAssetXml">
  <cs:param name="source">
    <asset name="First test asset" type="temp."/>
  </cs:param>
</cs:command>
```

# XSLT commands

censhare

- Asset management
  - Check out asset
    - Check out asset with given asset XML or ID
    - Attributes
      - “name”: “ com.censhare.api.assetmanagement.CheckOut”
      - “returning”: XML of checked out asset
    - Parameters
      - “source”: asset XML or asset ID to checkout
    - Example

```
<xsl:variable name="resultAssetXml"/>
<cs:command name="com.censhare.api.assetmanagement.CheckOut" returning="resultAssetXml">
  <cs:param name="source">
    <asset id="211421" currversion="0"/>
  </cs:param>
</cs:command>
```

# XSLT commands

censhare

- Asset management
  - Check in asset
    - Check in asset with given asset XML or ID
    - Attributes
      - “name”: “ com.censhare.api.assetmanagement.CheckIn”
      - “returning”: XML of checked in asset
    - Parameters
      - “source”: asset XML or asset ID to checkin
    - Example

```
<xsl:variable name="resultAssetXml"/>
<cs:command name="com.censhare.api.assetmanagement.CheckIn" returning="resultAssetXml">
  <cs:param name="source">
    <asset id="211421" currversion="0"/>
  </cs:param>
</cs:command>
```

# XSLT commands

censhare

- Asset management
  - Check out abort asset
    - Check out abort asset with given asset XML or ID
    - Attributes
      - “name”: “ com.censhare.api.assetmanagement.CheckOutAbort”
      - “returning”: XML of checked out abort asset
    - Parameters
      - “source”: asset XML or asset ID to check out abort
    - Example

```
<xsl:variable name="resultAssetXml"/>
<cs:command name="com.censhare.api.assetmanagement.CheckOutAbort" returning="resultAssetXml">
  <cs:param name="source">
    <asset id="211421"/>
  </cs:param>
</cs:command>
```

# XSLT commands

censhare

- Asset management
  - Update asset
    - Update asset with given asset XML (update in same asset version)
    - Attributes
      - “name”: “ com.censhare.api.assetmanagement.Update”
      - “returning”: XML of updated asset
    - Parameters
      - “source”: complete asset XML to update

# XSLT commands

censhare

- Asset management
  - Update asset
    - Example

```
<xsl:variable name="sourceAssetXml">
  <xsl:copy-of select="asset"/>
</xsl:variable>
<xsl:variable name="resultAssetXml"/>
<cs:command name="com.censhare.api.assetmanagement.Update" returning="resultAssetXml">
  <cs:param name="source">
    <asset>
      <xsl:copy-of select="$sourceAssetXml/asset/@*"/>
      <xsl:attribute name="name" select="'Test'"/>
      <xsl:copy-of select="$sourceAssetXml/asset/node()"/>
    </asset>
  </cs:param>
</cs:command>
```

# XSLT commands

censhare

- Asset management
  - Clone and clean asset XML
    - Clone and clean asset XML with given asset XML or ID (can be used to check in new asset)
    - Attributes
      - “name”: “ com.censhare.api.assetmanagement.CloneAndCleanAssetXml”
      - “returning”: XML of cloned and cleaned asset
    - Parameters
      - “source”: complete asset XML to clone and clean
    - Example
      - <xsl:variable name=”resultAssetXml”/>
        <cs:command name=”com.censhare.api.assetmanagement.CloneAndCleanAssetXml” returning=”resultAssetXml”>
        <cs:param name=”source” select=”asset”/>
        </cs:command>

# XSLT commands

censhare

- Asset management
  - Delete asset (censhare versions 4.5.22 4.6.22, 4.7.15, 4.8.0 or higher)
    - Delete asset with given asset XML (update in same asset version)
    - Attributes
      - “name”: “ com.censhare.api.assetmanagement.Delete”
      - “returning”: XML of updated asset
    - Parameters
      - “source”: asset XML or asset ID to delete
      - “state”: Possible values are “none” (for not deleted), “proposed” (for proposed for deletion), “marked” (for marked for deletion or “physical” (for physical deletion). This parameter is optional. If omitted the asset is proposed for deletion

# XSLT commands

censhare

- Asset management
  - Delete asset
    - Example

```
<cs:command name="com.censhare.api.assetmanagement.Delete">
  <cs:param name="source" select=“123456"/>
  <cs:param name="state" select=“'proposed'"/>
</cs:command>
```

# XSLT commands

censhare

- Asset management
  - Example: add selected asset to basket

```
<!-- get basket -->
<xsl:variable name="basketAsset" select="cs:asset()[@censhare:asset.type='order.shoppingbasket.'
  and @censhare:asset.wf_target=system-property('censhare:party-id')]"/>
<!-- create relation to basketAsset -->
<xsl:if test="$basketAsset and empty($basketAsset/child_asset_rel[@key='user.' and @child_asset=current()/@id])">
  <xsl:variable name="resultAssetXml"/>
  <cs:command name="com.censhare.api.assetmanagement.Update" returning="resultAssetXml">
    <cs:param name="source">
      <asset>
        <xsl:copy-of select="$basketAsset/@* | $basketAsset/node()"/>
        <child_asset_rel key="user." child_asset="{@id}"/>
      </asset>
    </cs:param>
  </cs:command>
</xsl:if>
```

# XSLT commands

censhare

- Asset management
  - Example: remove selected asset from basket

```
<!-- get basket -->
<xsl:variable name="basketAsset" select="cs:asset()[@censhare:asset.type='order.shoppingbasket.'
  and @censhare:asset.wf_target=system-property('censhare:party-id')]"/>
<!-- remove relation from basketAsset -->
<xsl:if test="$basketAsset">
  <xsl:variable name="resultAssetXml"/>
    <cs:command name="com.censhare.api.assetmanagement.Update" returning="resultAssetXml">
      <cs:param name="source">
       <asset>
         <xsl:copy-of select="$basketAsset/@* | $basketAsset/node()[not(node-name(.)='child_asset_rel'
           and @key='user.' and @child_asset=current()/@id)]"/>
       </asset>
    </cs:param>
  </cs:command>
</xsl:if>
```

# XSLT commands

censhare

- Transformation
  - Commands
    - XSL transformation
    - Asset transformation
    - Image transformation
    - Barcode transformation
    - FOP transformation

# XSLT commands

censhare

- Transformation
  - XSL transformation
    - Execute a XSL transformation (sub transformation)
    - Attributes
      - “name”: “com.censhare.api.transformation.XslTransformation”
      - “returning”: Result of the transformation (optional)
    - Parameters
      - “source”: The input XML to be transformed as XML, URL, file locator or storage item (required)
      - “dest”: The file locator of the destination file where the result of the transformation should be stored (optional)
        If this parameter is missing, the result is returned as result of this cs:command function
      - “stylesheet”:  Source of the stylesheet XML as XML, URL or file locator (required)
      - “xsl-parameters”:  Tunnel nested parameters directly to the XSLT (optional). Nested parameters can be XML nodes or simple values (e.g. string, number)

# XSLT commands

censhare

- Transformation
  - XSL transformation
    - Example

```
<cs:command name="com.censhare.api.transformation.XslTransformation" returning="dummy">
  <cs:param name="stylesheet" select="concat($config, 'modules/content_export/stylesheets/export.xsl'"/>
  <cs:param name="source" select="$chapterAsset/storage_item[@key='master'][1]"/>
  <cs:param name="dest" select="concat($parentDir, 'chapter-', $chapterAsset/@id, '.xhtml')"/>
  <cs:param name="xsl-parameters">
    <cs:param name="format" select="'xhtml'"/>
    <cs:param name="debug" select="true()"/>
  </cs:param>
</cs:command>
```

# XSLT commands

censhare

- Transformation
  - Asset transformation
    - Executes an asset transformation (sub transformation)
    - Same as “XSL transformation”, but stylesheet is used from a resource asset
    - Attributes
      - “name”: “com.censhare.api.transformation.AssetTransformation”
      - “returning”: Result of the transformation (optional)
    - Parameters
      - “source”: The input XML to be transformed as XML, URL, file locator or storage item (required)
      - “dest”: The file locator of the destination file where the result of the transformation should be stored (optional)
        If this parameter is missing, the result is returned as result of this cs:command function
      - “key”:  Key to the stylesheet resource asset (required)
      - “xsl-parameters”:  Tunnel nested parameters directly to the XSLT (optional)

# XSLT commands

censhare

- Transformation
  - Asset transformation
    - Example

```
<cs:command name="com.censhare.api.transformation.AssetTransformation" returning="dummy">
  <cs:param name="key" select="'censhare:epub-chapter'"/>
  <cs:param name="source" select="$chapterAsset/storage_item[@key='master'][1]"/>
  <cs:param name="dest" select="concat($parentDir, 'chapter-', $chapterAsset/@id, '.xhtml')"/>
  <cs:param name="xsl-parameters">
    <cs:param name="format" select="'xhtml'"/>
  </cs:param>
</cs:command>
```

# XSLT commands

censhare

- Transformation
  - Image transformation
    - Executes one or more image transformations (e.g. crop, scale, …)
    - Attributes
      - “name”: “com.censhare.api.transformation.ImageTransformation”
      - “returning”: Result of the transformation (optional)
    - Parameters
      - “source”: The input source (image) to be transformed (required). It must reference a storage item or a file
      - “dest”: The file locator of the destination file where the result of the transformation should be stored (required). It must reference a file in a virtual file system. The file extension is used to define the target image format (e.g. “a.jpg”)
      - “image-transformation”:  XML node, which contains the image transformations and transformation attributes (required)

# XSLT commands

censhare

- Transformation
  - Image transformation
    - Operations
      - Scale: <scale width="220" height="220"/>
        Crop: <crop x="100" y="20" width="220" height="220"/>
        Rotate: <rotate degree="90"/>
        Convert: <convert/>
        OPI: <opi/>
        Flip horizontally: <fliph/>
        Flip vertically: <flipv/>
        Overlay: <overlay overlay-image="cs:file-locator($overlayAsset/storage_item[@key='master'])" opacity="50" gravity="center" tile="false" extra-opts=""/>
        Video frame: <videoframe timecode="0:0:0.000"/>
        Clip (Use an embedded clipping path to produce a new image with a transparent alpha channel): <clip/>
        Frame (Scale input image proporionally to fit into a frame of a given width and height): <frame upscale="false" width="100" height="100" gravity="center" clip="false"/>

# XSLT commands

censhare

- Transformation
  - Image transformation
    - Example

```
<cs:command name="com.censhare.api.transformation.ImageTransformation">
  <cs:param name="source" select="$currentAsset/storage_item[@key='master']"/>
  <cs:param name="dest" select="concat($dir, 'imageTransformation/image.jpg')"/>
  <cs:param name="image-transformation">
    <image-transformation>
      <operations>
        <crop x="100" y="20" width="220" height="220"/>
        <rotate degree="90"/>
        <clip/>
      </operations>
      <attributes sharpen="true" out-jpeg-quality="100" out-color="cmyk"/>
    </image-transformation>
  </cs:param>
</cs:command>
```

# XSLT commands

censhare

- Transformation
  - Barcode transformation
    - Executes a barcode transformation
    - Used fonts must be available at startup of the JavaVirtualMachine
    - Attributes
      - “name”: “com.censhare.api.transformation.BarcodeTransformation”
    - Parameters
      - “source”: The input value to be represented by the barcode
      - “dest”: The file locator of the destination file where the result of the transformation should be stored. It must reference a file in a virtual file system (required)
      - “barcode-transformation”:  XML node, which contains the barcode transformations (required)

# XSLT commands

censhare

- Transformation
  - Barcode transformation
    - Parameter
      - “type”: ean8, ean13, qr (further types possible but not implemented yet)
        “file-format”: pdf, eps, svg, png
        “height”: mm value
        “module-width”: space between “bars”, leave empty for an automatic default value
        “margin”: mm value
        “orientation”: 0, 90, 180, 270 (degrees rotation)
        “font”: font name must be available to the JVM at server startup time
        “font-size”: pt value
        “qr-error-correction”: “”, “H”, “L”, “M”, “Q”
        “dpi”: dots per inch for rasterized image formats
        “antialias”: “true”, “false” for font antialiasing

# XSLT commands

censhare

- Transformation
  - Barcode transformation
    - Example

```
<cs:command name="com.censhare.api.transformation.BarcodeTransformation">
  <cs:param name="source" select="$source"/>
  <cs:param name="dest" select="$resultFile"/>
  <cs:param name="barcode-transformation">
    <barcode-transformation type="ean13" file-format="pdf" height="30" module-width="1" margin="4"
      orientation="0" font="Courier" font-size="6" qr-error-corr="" dpi="300" antialias="true"/>
  </cs:param>
</cs:command>
```

# XSLT commands

censhare

- Transformation
  - FOP transformation
    - Executes a FOP transformation (sub transformation)
    - Attributes
      - “name”: “com.censhare.api.transformation.FopTransformation”
      - “returning”: Result of the transformation (optional)
    - Parameters
      - “source”: The input XML to be transformed as XML, URL, file locator or storage item (required)
      - “dest”: The file locator of the destination file where the generated PDF should be stored. It must reference a file in a virtual file system (optional)
      - “stylesheet”:  Source of the stylesheet XML as XML, URL or file locator (required)
      - “xsl-parameters”:  Tunnel nested parameters directly to the XSLT (optional). The attributes “author”, “title”, “pdf-base-url”, “keywords”, “source-resolution”, “target-resolution” and “pdf-base14-kerning” are passed to the FOP processor

# XSLT commands

censhare

- Transformation
  - FOP transformation
    - Example

```
<cs:command name="com.censhare.api.transformation.FopTransformation" returning="dummy">
  <cs:param name="stylesheet"
    select="'censhare:///service/assets/asset;censhare:resource-key=test/storage/master/file'"/>
  <cs:param name="source" select="asset/storage_item[@key='master'][1]"/>
  <cs:param name="dest" select="concat($parentDir, 'chapter-', $chapterAsset/@id, '.pdf')"/>
  <cs:param name="xsl-parameters">
    <cs:param name="title" select="asset/@name"/>
    <cs:param name="asset" select="asset"/>
  </cs:param>
</cs:command>
```

# XSLT commands

censhare

- Office
  - Commands
    - Read Excel
    - Read Word

# XSLT commands

censhare

- Office
  - Read Excel
    - Read Excel document as XML
    - Attributes
      - “name”: “com.censhare.api.Office.ReadExcel”
      - “returning”: Result of the transformation (optional)
    - Parameters
      - “source”: The input XML to be transformed as XML, URL, file locator or storage item (required)
      - “dest”: The file locator of the destination file. It must reference a file in a virtual file system (optional)
      - “max-columns”: Maximal number of columns to read (optional, default is unlimited)
      - “max-rows”: Maximal number of rows to read (optional, default is unlimited)
      - “level”: “info” reads only information about sheets, styles and print areas. “full” reads all (optional, default is “full”)
      - “sheet-index”: reads only sheet with given index (optional)

# XSLT commands

censhare

- Office
  - Read Excel
    - Example

```
<cs:command name="com.censhare.api.Office.ReadExcel">
 <cs:param name="source" select="asset/storage_item[@key='master'][1]"/>
</cs:command>
```

# XSLT commands

censhare

- Office
  - Read Word
    - Read Word document as XML
    - Attributes
      - “name”: “com.censhare.api.Office.ReadWord”
      - “returning”: Result of the transformation (optional)
    - Parameters
      - “source”: The input XML to be transformed as XML, URL, file locator or storage item (required)
      - “dest”: The file locator of the destination file. It must reference a file in a virtual file system (optional)

# XSLT commands

censhare

- Office
  - Read Word
    - Example

```
<cs:command name="com.censhare.api.Office.ReadWord">
 <cs:param name="source" select="asset/storage_item[@key='master'][1]"/>
</cs:command>
```

# XSLT commands

censhare

- Context
  - Commands
    - Set property

# XSLT commands

censhare

- Context
  - Set property
    - Set static properties (variables) in global context
    - Attributes
      - “name”: “com.censhare.api.context.SetProperty”
    - Parameters
      - “name”: Name of the property with values
        - “censhare:command-xml”: read only access to command XML, if transformation was started in a command (server Action, asset automation, content export, transformation variant, ...)
        - “censhare:auto-delete”: set auto delete of working directory to “true” or “false”
        - “censhare:result-file-locator”: set result file to given file locator (Deprecated, use “cs:output”)
        - “censhare:result-mimetype”: set result mime type (Deprecated, use “cs:output”)
      - “value”: Value of the property

# XSLT commands

censhare

- Context
  - Set property
    - Example
      - Disable deletion of the working directory at end of transaction

```
<cs:command name="com.censhare.api.context.SetProperty">
  <cs:param name="name" select="'censhare:auto-delete'"/>
  <cs:param name="value" select="'false'"/>
</cs:command>
```

      - Set result file and result mimetype

```
<cs:command name="com.censhare.api.context.SetProperty">
  <cs:param name="name" select="'censhare:result-file-locator'"/>
  <cs:param name="value" select="$destFile"/>
</cs:command>
<cs:command name="com.censhare.api.context.SetProperty">
  <cs:param name="name" select="'censhare:result-mimetype'"/>
  <cs:param name="value" select="$sourceStorage/@mimetype"/>
</cs:command>
```

# XSLT commands

censhare

- I/O
  - Commands
    - Write XML
    - Read XML
    - Copy
    - Create virtual file system
    - Close virtual file system
    - Get file system reference

# XSLT commands

censhare

- I/O
  - Write XML
    - Writes XML to a file
    - Attributes
      - “name”: “com.censhare.api.io.WriteXML”
    - Parameters
      - “source”: The XML to be written (required)
      - “dest”: The file locator of the destination file (required). It must reference a file in a virtual file system
      - “output” An XML element “output” providing various serialization properties like pretty printing or encoding (optional). This element is identical to the XSLT output function (xsl:output) and supports most of it‘s attributes

# XSLT commands

censhare

- I/O
  - Write XML
    - Parameter
      - name? = qname
        method? = “xml” | “html” | “xhtml” | “text” | qname-but-not-ncname
        byte-order-mark? = “yes” | “no”
        cdata-section-elements? = qnames
        doctype-public? = string
        doctype-system? = string
        encoding? = string
        escape-uri-attributes? = “yes” | “no”
        include-content-type? = “yes” | “no”
        indent? = “yes” | “no”
        media-type? = string
        normalization-form? = “NFC” | “NFD” | “NFKC” | “NFKD” | “fully-normalized” | “none” | nmtoken
        omit-xml-declaration? = “yes” | “no”
        standalone? = “yes” | “no” | “omit”
        undeclare-prefixes? = “yes” | “no”
        version? = nmtoken

# XSLT commands

censhare

- I/O
  - Write XML
    - Example

```
<cs:command name=”com.censhare.api.io.WriteXML”>
  <cs:param name=”source” select=”$metaDoc”/>
  <cs:param name=”dest” select=”concat($out, 'EPUB-Debug.xml')”/>
  <cs:param name=”output”>
    <output indent=”yes”/>
  </cs:param>
</cs:command>
```

# XSLT commands

censhare

- I/O
  - Read XML
    - Reads XML content from a file or a storage item
    - Attributes
      - “name”: “com.censhare.api.io.ReadXML”
      - “returning”: readed XML (optional)
    - Parameters
      - “source”: The input source where to read from (file or storage item)

# XSLT commands

censhare

- I/O
  - Read XML
    - Example

```
<cs:command name=”com.censhare.api.io.ReadXML” returning=”content”>
  <cs:param name=”source” select=”asset/storage_item[@key='master'][1]”/>
</cs:command>
```

# XSLT commands

censhare

- I/O
  - Copy file
    - Copies a file or a storage item into a destination file
    - Attributes
      - “name”: “com.censhare.api.io.Copy”
    - Parameters
      - “source”: The input source to be copied (file or storage item)
      - “dest”: The file locator of the destination file. It must reference a file in a virtual file system

# XSLT commands

censhare

- I/O
  - Copy file
    - Example
      - Copy a storage item

```
<cs:command name="com.censhare.api.io.Copy">
  <cs:param name="source" select="asset/storage_item[@key='master'][1]"/>
  <cs:param name="dest" select="concat($out, 'result/image.jpg')"/>
</cs:command>
```

      - Copy a file

```
<cs:command name="com.censhare.api.io.GetFilesystemRef" returning="config">
  <cs:param name="name" select="'config-runtime'"/>
</cs:command>
<cs:command name="com.censhare.api.io.Copy">
  <cs:param name="source" select="concat($config, 'modules/resources/logo.jpg'"/>
  <cs:param name="dest" select="concat($out, 'result/image.jpg')"/>
</cs:command>
```

# XSLT commands

censhare

- I/O
  - Create virtual file system
    - Creates a (possibly virtual) directory, which can be used by the caller to write files into it
    - Attributes
      - “name”: “com.censhare.api.io.CreateVirtualFileSystem”
      - “returning”: The URL identifying the virtual file system. This string can (and must) be used for all subsequent output file references
    - Parameters
      - “format”: The result format of the file system (optional). If the file system should be packed into an archive, this parameter must be specified. Currently the only supported format is ZIP (value “zip”). If the parameter is missing, the result will be a directory structure

# XSLT commands

censhare

- I/O
  - Create virtual file system
    - Example

```
<cs:command name="com.censhare.api.io.CreateVirtualFileSystem" returning="out">
  <cs:param name="format" select="'zip'"/>
</cs:command>
```

# XSLT commands

censhare

- I/O
  - Close virtual file system
    - Closes a virtual file system
    - Attributes
      - “name”: “com.censhare.api.io.CloseVirtualFileSystem”
      - “returning”: The URL identifying the resulting directory or archive file. It can be used as a “censhare:result-file-locator” return parameter
    - Parameters
      - “id”: The file system reference as returned in “CreateVirtualFileSystem”
      - “name”: The name of the result archive file (optional). This parameter is ignored if the file system is either no ZIP archive or the parameter “dest” is specified
      - “dest”: A file reference where the archive file should be stored (optional). This parameter is ignored if the file system isn’t an archive

# XSLT commands

censhare

- I/O
  - Close virtual file system
    - Example

```
<cs:command name="com.censhare.api.io.CreateVirtualFileSystem" returning="out">
  <cs:param name="format" select="'zip'"/>
</cs:command>
```

```
<cs:command name="com.censhare.api.io.CloseVirtualFileSystem" returning="zip">
  <cs:param name="id" select="$out"/>
  <cs:param name="name" select="concat(asset/@name, '.zip')"/>
</cs:command>
```

# XSLT commands

censhare

- I/O
  - Get file system reference
    - Get a reference to a server file system for reading only
      - Returns a reference to the named file system, which must be a server file system. This file system can be used for reading only. It is not possible to write into it.
        Currently the only supported file system is the server's configuration (“config-runtime”). That way it is possible to include additional resources like config files or XSLT stylesheets, which are not stored as resource assets.
    - Attributes
      - “name”: “com.censhare.api.io.GetFilesystemRef”
      - “returning”: The URL identifying the resulting directory or archive file. It can be used as a “censhare:result-file-locator” return parameter
    - Parameters
      - “name”: The name of the file system

# XSLT commands

censhare

- I/O
  - Get file system reference
    - Example

```
<!-- get file system reference of config runtime -->
<cs:command name="com.censhare.api.io.GetFilesystemRef" returning="config">
  <cs:param name="name" select="'config-runtime'"/>
</cs:command>
<!-- execute transformation with stylesheet of config runtime file system -->
<cs:command name="com.censhare.api.transformation.XslTransformation" returning="xml">
  <cs:param name="stylesheet" select="concat($config, 'modules/content_export/stylesheets/export.xsl'"/>
  <cs:param name="source" select="asset/storage_item[@key='master'][1]"/>
</cs:command>
```

# XSLT commands

censhare

- Renderer
  - Commands
    - Render

# XSLT commands

censhare

- Renderer
  - Render
    - Executes the rendering commands given through the Render-Client
    - Same syntax of XML message exchanging between censhare Server and Render-Client
    - Attributes
      - “name”: “com.censhare.api.renderer.Render”
      - “returning”: The URL identifying the resulting directory or archive file. It can be used as a “censhare:result-file-locator” return parameter
    - Parameters
      - “instructions”: XML structure with render instructions
      - “facility”: String, currently allowed values: “indesign-6.0” for CS-4, “indesign-7.0” for CS-5, “indesign-7.5” for CS-5.5, “indesign-8.0” for CS-6
      - “timeout”: timeout in seconds (default is 120 seconds)

# XSLT commands

censhare

- Renderer
  - Render
    - Methods
      - check-missing-fonts, check-placed-collections, close, copy-pagearea, copy-pagearea-asset, correct-document, debug, detach, detach-pagearea, edit-placement, eps, mark-not-printable-boxes, move-box, move-pagearea, open, pdf, place, place-collection, preview, rename-group, save, script, text-content, update-asset-element-structure, update-auto-client-variant, update-content, update-geometry, update-placeholder,

# XSLT commands

censhare

- Renderer
  - Render
    - Example

```
<cs:command name="com.censhare.api.renderer.Render">
  <cs:param name="facility" select="{concat('indesign-', $indesignStorageItem/@app_version)}"/>
  <cs:param name="instructions">
    <cmd>
      <renderer>
        <command method="open" asset_id="{asset[1]/@id}" document_ref_id="1"/>
        <command method="preview" spread="false" asset_element_id="1" document_ref_id="1"/>
        <command method="pdf" spread="false" document_ref_id="1"/>
        <command method="close" document_ref_id="1"/>
      </renderer>
      <assets>
        <xsl:copy-of select="$asset"/>
      </assets>
    </cmd>
  </cs:param>
</cs:command>
```

# XSLT commands

censhare

- PDF
  - Commands
    - Combine PDF
    - Get info of PDF
    - Get data of PDF forms

# XSLT commands

censhare

- PDF
  - Combine PDF
    - Combines given PDF sources to one PDF
    - Attributes
      - “name”: “com.censhare.api.pdf.CombinePDF”
    - Parameters
      - “dest”: A file reference where the archive file should be stored. This parameter is ignored if the file system isn’t an archive
      - “sources”: XML structure with PDF sources
        - “source” element
          - “href” attribute: The URL where the PDF document can be found. Can be a censhare URL or a standard URL (“href://www…”)
          - “pages” attribute: A list of page numbers (optional). Can be a figure for single page (e.g. “5”) or a list of pages (e.g. “(1,2,3)”). If not “pages” attribute exists, then all pages are used. If a page number not exists in a document, then this page is not used.

# XSLT commands

censhare

- PDF
  - Combine PDF
    - Example

```
<xsl:variable name="renderResult">
  <cs:command name="com.censhare.api.pdf.CombinePDF">
    <cs:param name="dest" select="$resultFile"/>
    <cs:param name="sources">
      <source href="file:/modules/test/test.pdf" pages="(1,2,3)"/>
      <source href="http://www.censhare.com/_storage/19623/122939/censhare+Feature+Summary.pdf"/>
      <source href="censhare:///service/assets/asset/id/203917/storage/master/file/145053.pdf"/>
      <source href="censhare:///service/filesystem/temp/_69e9d2f4-9038-4b24-9840-2d72266d64af/1.pdf"/>
    </cs:param>
  </cs:command>
</xsl:variable>
```

# XSLT commands

censhare

- PDF
  - Get info of PDF (censhare 4.8 or higher)
    - Get info of given PDF source
      - Pages with sizes
      - Notes with position and content
      - …
    - Attributes
      - “name”: “com.censhare.api.pdf.InfoPDF”
      - “returning”: The information about all objects in the given PDF document as XML (optional)
    - Parameters
      - “source”: The URL where the PDF document can be found. Can be a censhare URL or a standard URL (”href://www…”)

# XSLT commands

censhare

- PDF
  - Get info of PDF
    - Example

```
<cs:command name="com.censhare.api.pdf.InfoPDF">
  <cs:param name="source"
    select="'http://www.censhare.com/_storage/19623/122939/censhare+Feature+Summary.pdf'">
  </cs:param>
</cs:command>
```

    - Result

```
<?xml version="1.0" encoding="UTF-8"?>
<InfoPDF version="1.0">
  <Catalog>
    <AcroForm object-number="2" object-generation="0">
      <AcroForm>
        <DA>/Helv 0 Tf 0 g </DA>
        <DR>
          <Encoding>
            <PDFDocEncoding object-number="57" object-generation="0">
              <PDFDocEncoding>
                <Differences>
                  <item>24</item>
                  <item>breve</item>
                  <item>caron</item>
                  <item>circumflex</item>
```

# XSLT commands

censhare

- PDF
  - Get data of PDF forms (censhare 4.8 or higher)
    - Get data of the forms of given PDF source in the defined format
    - Attributes
      - “name”: “com.censhare.api.pdf.formPDF”
      - “returning”: The information about forms of the given PDF document in the defined format (optional)
    - Parameters
      - “source”: The URL where the PDF document can be found. Can be a censhare URL or a standard URL (“href://www…”)
      - “type”: The format of the result. Default value is “XFDF”. Possible values are:
        - “FDF”: Forms Data Format
        - “XFDF”: XML Forms Data Format
        - “XFA”: XML Forms Architecture

# XSLT commands

censhare

- PDF
  - Get data of PDF forms
    - Example

```
<cs:command name="com.censhare.api.pdf.FormPDF">
  <cs:param name="source"
    select="'censhare:///service/assets/asset/id/203917/storage/master/file/145053.pdf'">
  </cs:param>
</cs:command>
```

    - Result

```
<?xml version="1.0" encoding="UTF-8"?>
<form-data type="XFDF">
  <xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
    <ids original="16449DB8CB323843F2F1CB2033C0242D" modified="B392B9652234447B9CDFE0740F491CDA"/>
    <fields>
      <field name="LastName">
        <value>Content for last name</value>
      </field>
      <field name="FirstName">
        <value>Content for first name</value>
      </field>
      <field name="MiddleName">
        <value>Content for middle name</value>
      </field>
    </fields>
```

# XSLT commands

censhare

- Social networks
  - Commands
    - Twitter
      - Search messages
      - Read messages
      - Post message
    - Facebook
      - Get friends
      - Search
      - Get info
      - Post link, message, photo, video
    - YouTube
      - Get video info
      - List video categories
      - Upload video

# XSLT commands

censhare

- Social networks
  - Search Twitter messages (censhare 4.8 or higher)
    - Search for twitter messages, that contains the given text
    - Attributes
      - “name”: “com.censhare.api.twitter.search”
      - “returning”: The found Twitter messages as XML (optional)
    - Parameters
      - “meta-data”: XML with following structure
        - <account accessToken=”123456” accessTokenSecret=”123456”/> Please replace with your account info
          <query>censhare</query> Text to search
          <max-id>335356243097034753</max-id> Returns messages equal or less given ID (optional)
          <since-id>335356243097034700</since-id> Returns messages greater than given ID (optional)
          <count>20</count> Maximal messages (default=20, min=1, max=200, optional)
          <lang>de</lang> Restricts tweets to the given language (optional)

# XSLT commands

censhare

- Social networks
  - Search Twitter messages
    - Example (search for the first 20 messages, that contains “censhare” in german language)

```
<cs:command name="com.censhare.api.twitter.search">
  <cs:param name="meta-data">
    <account accessToken="123456-123456" accessTokenSecret="123456-123456"/>
    <query>censhare</query>
    <count>20</count>
  </cs:param>
</cs:command>
```

    - Result

```
<?xml version="1.0" encoding="UTF-8"?>
<_object_>
  <statuses created_at="Tue Oct 08 12:07:23 +0000 2013" text="RT @miguel_libro: Hecho con censhare: recetas de cocina para
    <metadata result_type="recent" iso_language_code="es"/>
    <user id="267626688" screen_name="test usert" name="Test User" id_str="267626688" location="Romandie" protected="false
      <entities>
        <url>
          <urls url="http://t.co/kAqrI5wkXz" indices="22" display_url="test.ch" expanded_url="http://www.test.ch"/>
        </url>
        <description/>
      </entities>
    </user>
    <retweeted_status created_at="Tue Oct 08 09:05:37 +0000 2013" text="Hecho con censhare: recetas de cocina para papel,
```

# XSLT commands

censhare

- Social networks
  - Read Twitter messages (censhare 4.8 or higher)
    - Read messages from the given Twitter account
    - Attributes
      - “name”: “com.censhare.api.twitter.ReadMessages”
      - “returning”: The found Twitter messages as XML (optional)
    - Parameters
      - “meta-data”: XML with following structure
        - <account accessToken=”123456” accessTokenSecret=”123456”/>
          <max-id>335356243097034753</max-id>
          <count>20</count>

Please replace with your account info
Returns messages greater than given ID (optional)
Maximal messages (default=20, min=1, max=200, optional)

# XSLT commands

censhare

- Social networks
  - Read Twitter messages
    - Example (read the first 20 messages)

```
<cs:command name="com.censhare.api.twitter.ReadMessages">
  <cs:param name="meta-data">
    <account accessToken="123456-123456" accessTokenSecret="123456-123456"/>
    <count>20</count>
  </cs:param>
</cs:command>
```

    - Result

```
<?xml version="1.0" encoding="UTF-8"?>
<_array_>
  <_object_ created_at="Thu Oct 10 06:06:47 +0000 2013" text=""Test for a twitter message" http://t.co/LQonCJKlDv" id_str=
    <user id="61132188" screen_name="test" name="Test" id_str="61132188" location="Germany - Baden-Baden" protected="false
      <entities>
        <url>
          <urls url="http://t.co/ZhY8SzKJxs" indices="22" display_url="test.com" expanded_url="http://www.test.com"/>
        </url>
        <description/>
      </entities>
    </user>
    <entities>
      <urls url="http://t.co/LQonCJKlDv" indices="50" display_url="bit.ly/1bdPVUq" expanded_url="http://bit.ly/1bdPVUq"/>
```

# XSLT commands

censhare

- Social networks
  - Post Twitter message (censhare 4.8 or higher)
    - Post given Twitter message with given account
    - Attributes
      - “name”: “com.censhare.api.twitter.PostMessage”
      - “returning”: The posted Twitter message as XML (optional)
    - Parameters
      - “meta-data”: XML with following structure
        - <account accessToken=”123456” accessTokenSecret=”123456”/> Please replace with your account info
          <message>First message created with censhare API</message> Message to post

# XSLT commands

censhare

- Social networks
  - Post Twitter message
    - Example

```
<cs:command name="com.censhare.api.twitter.PostMessage">
  <cs:param name="meta-data">
    <account accessToken="123456-123456" accessTokenSecret="123456-123456"/>
    <message>Message created with censhare API</message>
  </cs:param>
</cs:command>
```

    - Result

```
<?xml version="1.0" encoding="UTF-8"?>
<_object_ created_at="Thu Oct 10 07:00:26 +0000 2013" text="Message created with censhare API" id_str="388197314764214272
  <user id="53875558" screen_name="test" name="Test" id_str="53875558" location="" followers_count="28"
    <entities>
      <description/>
    </entities>
  </user>
  <entities/>
</_object_>
```

# XSLT commands

censhare

- Social networks
  - Get Facebook friends (censhare 4.8 or higher)
    - Get information about friends of given account
    - Attributes
      - “name”: “com.censhare.api.facebook.GetFriends”
      - “returning”: The information about friends as XML (optional)
    - Parameters
      - “meta-data”: XML with following structure
        - <account accessToken=”123456”/>

Please replace with your account info

# XSLT commands

censhare

- Social networks
  - Get Facebook friends
    - Example

```
<cs:command name="com.censhare.api.facebook.GetFriends">
  <cs:param name="meta-data">
    <account accessToken="123456-123456"/>
  </cs:param>
</cs:command>
```

    - Result

```
<?xml version="1.0" encoding="UTF-8"?>
<_object_>
  <data name="User 1" id="530483459"/>
  <data name="User 2" id="570876837"/>
  <data name="User 3" id="643939745"/>
  <data name="User 4" id="676991221"/>
  <data name="User 5" id="1365394659"/>
  <data name="User 6" id="1446067322"/>
  <data name="User 7" id="1471142286"/>
  <data name="User 8" id="1491175033"/>
  <data name="User 9" id="1503474893"/>
  <data name="User 10" id="100000094663111"/>
  <data name="User 11" id="100000397313822"/>
  <data name="User 12" id="100000094663113"/>
  <data name="User 13" id="100000397313824"/>
```

# XSLT commands

censhare

- Social networks
  - Search for Facebook objects (censhare 4.8 or higher)
    - Search for Facebook objects that matches with given parameters
    - For more information please visit “https://developers.facebook.com/docs/reference/api/search/”
    - Attributes
      - “name”: “com.censhare.api.facebook.Search”
      - “returning”: The information about the object as XML (optional)

# XSLT commands

censhare

- Social networks
  - Search for Facebook objects
    - Parameters
      - “query”: search term
      - “type”: result types. Values: post, user, page, event, group, place, checkin, location (default=”post”, optional)
      - “count”: maximal results (default=20, max=1000, optional)
      - “distance”: may limit search for “place” or “location” (optional)
      - “center”: may limit search for “place” or “location” (optional)
      - “place”: may limit search for “place” or “location” (optional)
      - “parts”: any string, if left out all possible fields are returned (optional)
      - “meta-data”: XML with following structure
        - <account accessToken=”123456”/>

Please replace with your account info

# XSLT commands

censhare

- Social networks
  - Search for Facebook objects
    - Example

```
<cs:command name="com.censhare.api.facebook.Search">
  <cs:param name="query" select="'censhare'"/>
  <cs:param name="type" select="'post'"/>
  <cs:param name="count" select="20"/>
  <cs:param name="distance" select="1000"/>
  <cs:param name="center" select="'37.76,-122.427'"/>
  <cs:param name="place" select="'166793820034304'"/>
  <cs:param name="parts" select="'id,name'"/>
  <cs:param name="meta-data">
    <account accessToken="123456-123456"/>
  </cs:param>
</cs:command>
```

    - Result

```
<?xml version="1.0" encoding="UTF-8"?>
<_object_>
  <data id="347808794754_10151755715814755" link="http://www.crossmedialive.com/Exhibitor/Censhare-UK-Limited"
    <from category="Software" id="347808794754" name="censhare"/>
    <actions name="Comment" link="https://www.facebook.com/347808794754/posts/10151755715814755"/>
    <actions name="Like" link="https://www.facebook.com/347808794754/posts/10151755715814755"/>
    <privacy value=""/>
```

# XSLT commands

censhare

- Social networks
  - Get information about Facebook object (censhare 4.8 or higher)
    - Get information about Facebook object with given ID
    - Attributes
      - “name”: “com.censhare.api.facebook.GetInfo”
      - “returning”: The information about a Facebook object as XML (optional)
    - Parameters
      - “id”: ID of Facebook object
      - “parts”: List of “fields”. If empty, then all fields are returned (optional)
      - “meta-data”: XML with following structure
        - <account accessToken=”123456”/>

Please replace with your account info

# XSLT commands

censhare

- Social networks
  - Get information about Facebook object
    - Example

```
<cs:command name="com.censhare.api.facebook.GetInfo">
  <cs:param name="id" select="'100001674305953'"/>
  <cs:param name="parts" select="'id,name'"/>
  <cs:param name="meta-data">
    <account accessToken="123456-123456"/>
  </cs:param>
</cs:command>
```

    - Result

```
<?xml version="1.0" encoding="UTF-8"?>
<_object_>
```

# XSLT commands

censhare

- Social networks
  - Post Facebook video (censhare 4.8 or higher)
    - Post a video to Facebook
    - Attributes
      - “name”: “com.censhare.api.facebook.PostVideo”
      - “returning”: The information about a Facebook object as XML (optional)
    - Parameters
      - “source”: The URL of the video. Can be a censhare URL or a standard URL (”href://www…”)
      - “meta-data”: XML with following structure
        - <account accessToken=”123456”/> Please replace with your account info
        - <title>title</title> Title of the video
        - <description>description</description> Description of the video (optional)

# XSLT commands

censhare

- Social networks
  - Post Facebook video
    - Example

```
<cs:command name="com.censhare.api.facebook.PostVideo">
  <cs:param name="source" select="$storageURL"/>
  <cs:param name="meta-data">
    <account accessToken="123456-123456"/>
    <title>Sample video</title>
    <description>A sample description for the uploaded video</description>
  </cs:param>
</cs:command>
```

    - Result

```
<?xml version="1.0" encoding="UTF-8"?>
<_object_ id="591184504280688"/>
```

# XSLT commands

censhare

- Social networks
  - Post Facebook photo (censhare 4.8 or higher)
    - Post a photo to Facebook
    - Attributes
      - “name”: “com.censhare.api.facebook.PostPhoto”
      - “returning”: The information about a Facebook object as XML (optional)
    - Parameters
      - “source”: The URL of the photo. Can be a censhare URL or a standard URL (“href://www…”)
      - “meta-data”: XML with following structure
        - <account accessToken=”123456”/> Please replace with your account info
        - <title>title</title> Title of the photo
        - <description>description</description> Description of the photo (optional)

# XSLT commands

censhare

- Social networks
  - Post Facebook photo
    - Example

```
<cs:command name="com.censhare.api.facebook.PostPhoto">
  <cs:param name="source" select="$storageURL"/>
  <cs:param name="meta-data">
    <account accessToken="123456-123456"/>
    <title>Sample video</title>
    <description>A sample description for the uploaded photo</description>
  </cs:param>
</cs:command>
```

    - Result

```
<?xml version="1.0" encoding="UTF-8"?>
<_object_ id="591184504280688"/>
```

# XSLT commands

censhare

- Social networks
  - Post Facebook message (censhare 4.8 or higher)
    - Post a message to Facebook
    - Attributes
      - “name”: “com.censhare.api.facebook.PostMessage”
      - “returning”: The information about a Facebook object as XML (optional)
    - Parameters
      - “meta-data”: XML with following structure
        - <account accessToken=”123456”/> Please replace with your account info
        - <message>message</message> Message to post

# XSLT commands

censhare

- Social networks
  - Post Facebook message
    - Example

```
<cs:command name="com.censhare.api.facebook.PostMessage">
  <cs:param name="meta-data">
    <account accessToken="123456-123456"/>
    <message>Sample message</message>
  </cs:param>
</cs:command>
```

    - Result

```
<?xml version="1.0" encoding="UTF-8"?>
<_object_ id="591184504280688"/>
```

# XSLT commands

censhare

- Social networks
  - Post Facebook link (censhare 4.8 or higher)
    - Post a link to Facebook
    - Attributes
      - “name”: “com.censhare.api.facebook.PostLink”
      - “returning”: The information about a Facebook object as XML (optional)
    - Parameters
      - “link”: The link to post
      - “meta-data”: XML with following structure
        - <account accessToken=”123456”/> Please replace with your account info
        - <message>message</message> Message to the link (optional)
        - <description>description</description> Description of the link (optional)
        - <caption>caption</caption> caption to the link (optional)

# XSLT commands

censhare

- Social networks
  - Post Facebook link
    - Example

```
<cs:command name="com.censhare.api.facebook.PostLink">
  <cs:param name="link" select="'http://www.censhare.com'"/>
  <cs:param name="meta-data">
    <account accessToken="123456-123456"/>
    <message>Sample message</message>
    <message>This message is a demonstration of an automated link post</message>
    <description>Some description text for posted link</description>
    <caption>A nice link caption</caption>
  </cs:param>
</cs:command>
```

    - Result

```
<?xml version="1.0" encoding="UTF-8"?>
<_object_ id="591184504280688"/>
```

# XSLT commands

censhare

- Social networks
  - Post YouTube video (censhare 4.8 or higher)
    - Post a video to YouTube
    - Attributes
      - “name”: “com.censhare.api.youtube.UploadVideo”
      - “returning”: The information about a YouTube object as XML (optional)
    - Parameters
      - “source”: The URL of the video. Can be a censhare URL or a standard URL (”href://www…”)
      - “meta-data”: XML with following structure
        - <account accessToken=”123456” refreshToken= “123456”/>
        - <json>parameters as JSON</json>

Please replace with your account info

Parameters as JSON. See more infos at https://developers.google.com/youtube/v3/docs/videos

# XSLT commands

censhare

- Social networks
  - Post YouTube video
    - Example

```
<cs:command name="com.censhare.api.facebook.PostLink">
  <cs:param name="source" select="$storageURL"/>
  <cs:param name="meta-data">
    <account accessToken="123456-123456" refreshToken="123456-123456"/>
    <json>
      {
        “snippet": {"title": “Video title", “description": “This is only a test"},
        “status": {"privacyStatus": “private"}
      }
    </json>
  </cs:param>
</cs:command>
```

    - Result

```
<?xml version="1.0" encoding="UTF-8"?>
<_object_ id="VSm7v_FIm_g" etag="&amp;quot;4IfXb8Dc78bqDKapqHxgOpqGpzM/pMaHpuxBNT0KgWwLOFQCF6vEsXQ&amp;quot;" kind="youtube#video"
  <snippet publishedAt="2013-04-24T05:02:19.000Z" description="this is only a test" title="Video title" channelId="UC4b8aF
    <thumbnails>
      <default url="https://i.ytimg.com/vi/VSm7v_FIm_g/default.jpg"/>
      <medium url="https://i.ytimg.com/vi/VSm7v_FIm_g/mqdefault.jpg"/>
      <high url="https://i.ytimg.com/vi/VSm7v_FIm_g/hqdefault.jpg"/>
```

# XSLT commands

censhare

- Social networks
  - Get info of YouTube video (censhare 4.8 or higher)
    - Get info of YouTube video with given ID
    - Attributes
      - “name”: “com.censhare.api.youtube.GetVideoInfo”
      - “returning”: The information about a YouTube object as XML (optional)
    - Parameters
      - “id”: The ID of the video
      - “parts”: any string. Possible values are id, snippet, contentDetails, fileDetails, player, processingDetails, recordingDetails, statistics, status, suggestions, topicDetails. If left out all possible fields are returned (optional)
      - “meta-data”: XML with following structure
        - <account accessToken=”123456” refreshToken= “123456”/>

Please replace with your account info

# XSLT commands

censhare

- Social networks
  - Get info of YouTube video
    - Example

```
<cs:command name="com.censhare.api.youtube.GetVideoInfo">
  <cs:param name="id" select="'UC4b8aRTxgUFaJiD9dn7bKwA'"/>
  <cs:param name="parts" select="'id,snippet,contentDetails'"/>
  <cs:param name="meta-data">
    <account accessToken="123456-123456" refreshToken="123456-123456"/>
  </cs:param>
</cs:command>
```

    - Result

```
<?xml version="1.0" encoding="UTF-8"?>
<_object_ kind="youtube#videoListResponse" etag="&quot;4IfXb8Dc78bqDKapqHxgOpqGpzM/-Ga6ckWUB4p8aT0SnAv0afDIa10&quot;">
  <items id="VSm7v_FIm_g" etag="&quot;4IfXb8Dc78bqDKapqHxgOpqGpzM/bxR0N_FyaaBNtEOEbrACTebzUiM&quot;" kind="youtube#video">
    <snippet publishedAt="2013-04-24T05:02:19.000Z" description="this is only a test" title="some video title" channelId="
      <thumbnails>
        <default url="https://i.ytimg.com/vi/VSm7v_FIm_g/default.jpg"/>
        <medium url="https://i.ytimg.com/vi/VSm7v_FIm_g/mqdefault.jpg"/>
        <high url="https://i.ytimg.com/vi/VSm7v_FIm_g/hqdefault.jpg"/>
      </thumbnails>
    </snippet>
    <contentDetails duration="PT3S" caption="false" definition="hd" dimension="2d" licensedContent="false"/>
```

# XSLT commands

censhare

- Social networks
  - Get YouTube video categories (censhare 4.8 or higher)
    - List all available video categories of YouTube to use e.g. in a user interface to select it
    - Attributes
      - “name”: “com.censhare.api.youtube.ListVideoCategories”
      - “returning”: The list of all categories as XML (optional)
    - Parameters
      - “region”: returns categories only for this region. See http://www.iso.org/iso/country_codes/iso_3166_code_lists/country_names_and_code_elements.htm for possible values
      - “language”: returns category names in this language (“en_US”, “de”, ...)
      - “meta-data”: XML with following structure
        - <account accessToken=”123456” refreshToken= “123456”/>

Please replace with your account info

# XSLT commands

censhare

- Social networks
  - Get YouTube video categories
    - Example

```
<cs:command name="com.censhare.api.facebook.ListVideoCategories">
  <cs:param name="region" select="'DE'"/>
  <cs:param name="lang" select="'DE'"/>
  <cs:param name="meta-data">
    <account accessToken="123456-123456" refreshToken="123456-123456"/>
  </cs:param>
</cs:command>
```

    - Result

```
<?xml version="1.0" encoding="UTF-8"?>
<_object_ kind="youtube#videoCategoryListResponse" etag="&quot;4IfXb8Dc78bqDKapqHxgOpqGpzM/KtIN9V7_rbKGFbFfL2QhyGHzREE&quo
  <items id="1" etag="&quot;4IfXb8Dc78bqDKapqHxgOpqGpzM/gljmmWdkbtlCrMi74KVYLrbTaIA&quot;" kind="youtube#videoCategory">
    <snippet channelId="UCBR8-60-B28hp2BmDPdntcQ" title="Film &amp; Animation"/>
  </items>
  <items id="2" etag="&quot;4IfXb8Dc78bqDKapqHxgOpqGpzM/0Hi2t81xRQV7OvkAVEYoTScHWGg&quot;" kind="youtube#videoCategory">
    <snippet channelId="UCBR8-60-B28hp2BmDPdntcQ" title="Autos &amp; Fahrzeuge"/>
  </items>
  <items id="10" etag="&quot;4IfXb8Dc78bqDKapqHxgOpqGpzM/vxKABLebu8Q503HPLEOmSyBvyks&quot;" kind="youtube#videoCategory">
    <snippet channelId="UCBR8-60-B28hp2BmDPdntcQ" title="Musik"/>
  </items>
```

# XSLT commands

censhare

- E-mail
  - Commands
    - Send mail
    - List mail
    - Read mail
    - Delete mail

# XSLT commands

censhare

- E-mail
  - Send e-mails (censhare 4.8 or higher)
    - Attributes
      - “name”: “com.censhare.api.Mail.SendMail”
      - “returning”: Sent timestamp as XML (optional)
    - Parameters
      - “source”: Definition of e-mails to send
        - <mails account=”account name”>
          - <mail sender-address=”sender address” subject=”subject”>
            - <recipient type=”to” address=”rm@censhare.de”/>
            - <content mimetype=”text/plain”>Content
              or
            - <multipart-body><content mimetype=”text/plain”>Content
            - <multipart-body filename=”file name” url=”url”/>

Defined e-mail account in e-mail service
Sender address and subject
Recipient (types=”to”, “cc”, “bcc”, default=”to”)
Content mimetype (default=”text/plain”)

# XSLT commands

censhare

- E-mail
  - Send e-mails
    - Example simple

```
<cs:command name="com.censhare.api.Mail.SendMail">
  <cs:param name="source">
    <mails account-name="corpus">
      <mail sender-address="Robert Motzke &lt;test@motzke.com&gt;" subject="Test mail">
        <recipient type="to" address="rm@censhare.de"/>
        <content mimetype="text/html" transfer-charset="UTF-8">
          <h3>Test</h3>
          <p>Test content</p>
        </content>
      </mail>
    </mails>
  </cs:param>
</cs:command>
```

    - Result

```
<?xml version="1.0" encoding="UTF-8"?>
<sent>2013-09-08 9:51:25 +0200</sent>
```

# XSLT commands

censhare

- E-mail
  - Send e-mails
    - Example multi part with attached file

```
<cs:command name="com.censhare.api.Mail.SendMail">
  <cs:param name="source">
    <mails account-name="corpus">
      <mail sender-address="Robert Motzke &lt;test@motzke.com&gt;" subject="Test with file">
        <recipient type="to" address="rm@censhare.de"/>
        <multipart-body>
          <content mimetype="text/plain" transfer-charset="UTF-8">
            <xsl:text>Test for plain text</xsl:text>
          </content>
        </multipart-body>
        <multipart-body>
          <content mimetype="text/html" transfer-charset="UTF-8">
            <h3>Test</h3>
            <p>Test content</p>
          </content>
        </multipart-body>
        <multipart-body filename="test.pdf" url="{@url}"/>
      </mail>
    </mails>
  </cs:param>
</cs:command>
```

# XSLT commands

censhare

- E-mail
  - List e-mails (censhare 4.8 or higher)
    - List e-mails of given account and given folder
    - Attributes
      - “name”: “com.censhare.api.Mail.ListMail”
      - “returning”: List of e-mails as XML (optional). E-mail with ID=1 is the oldest e-mail.
    - Parameters
      - “source”: E-mail account and folder to list
        - <mails account=”account name” folder=”INBOX”>

Defined e-mail account in e-mail service.
Folder is inbox folder (default=”INBOX”)

# XSLT commands

censhare

- E-mail
  - List e-mails
    - Example

```
<cs:command name="com.censhare.api.Mail.ListMail">
  <cs:param name="source">
    <mails account-name="corpus" folder="INBOX"/>
  </cs:param>
</cs:command>
```

    - Result

```
<?xml version="1.0" encoding="UTF-8"?>
<mails account-name="corpus" folder="INBOX" xmlns:my="http://www.censhare.com" xmlns:corpus="http://www.censhare.com/xml/3
  <mail id="1" replyto-address="Robert Motzke &lt;robert@motzke.com&gt;" from-address="Robert Motzke &lt;robert@motzke.com
    <recipient address="test@motzke.com" type="to"/>
  </mail>
  <mail id="2" replyto-address="Robert Motzke &lt;test@motzke.com&gt;" from-address="Robert Motzke &lt;test@motzke.com&gt;
    <recipient address="Robert Motzke &lt;test@motzke.com&gt;" type="to"/>
  </mail>
</mails>
```

# XSLT commands

censhare

- E-mail
  - Read e-mail (censhare 4.8 or higher)
    - Read e-mails of given account and IDs
    - Attributes
      - “name”: “com.censhare.api.Mail.ReadMail”
      - “returning”: List of e-mails as XML (optional)
    - Parameters
      - “source”: E-mail account and folder to list
        - <mails account=”account name” folder=”INBOX” show-envelope=”true”>
          - <mail id=”1” delete-after-reading=”false”/>

E-mail account and inbox folder
If true then containing all headers as attributes

Mail and ID

# XSLT commands

censhare

- E-mail
  - Read e-mails
    - Example

```
<cs:command name="com.censhare.api.Mail.ReadMail">
  <cs:param name="source">
    <mails account-name="corpus" folder="INBOX" show-envelope="false">
      <mail id="1" delete-after-reading="false"/>
    </mails>
  </cs:param>
</cs:command>
```

    - Result

```
<?xml version="1.0" encoding="UTF-8"?>
<mails account-name="corpus" folder="INBOX" show-envelope="false" xmlns:corpus="http://www.censhare.com/xml/3.0.0/corpus"
  <mail id="2" delete-after-reading="false" replyto-address="Robert Motzke &lt;test@motzke.com&gt;" from-address="Robert M
    <recipient address="Robert Motzke &lt;test@motzke.com&gt;" type="to"/>
    <content type="text/html; charset=UTF-8">&lt;content mimetype="text/html" transfer-charset="UTF-8"&gt;&lt;h3&gt;Test&l
    <content filename="171362.jpg" Content-Disposition="attachment; filename=171362.jpg" Content-Transfer-Encoding="base64
  </mail>
</mails>
```

# XSLT commands

censhare

- E-mail
  - Delete e-mail (censhare 4.8 or higher)
    - Delete e-mails of given account and IDs
    - Attributes
      - “name”: “com.censhare.api.Mail.DeleteMail”
      - “returning”: Confirmation as XML (optional)
    - Parameters
      - “source”: E-mail account and folder to delete e-mails
        - <mails account=”account name” folder=”INBOX”> E-mail account and inbox folder
          - <mail id=”1”/> Mail and ID

# XSLT commands

censhare

- E-mail
  - Delete e-mails
    - Example

```
<cs:command name="com.censhare.api.Mail.DeleteMail">
  <cs:param name="source">
    <mails account-name="corpus" folder="INBOX">
      <mail id="1"/>
    </mails>
  </cs:param>
</cs:command>
```

    - Result

```
<?xml version="1.0" encoding="UTF-8"?>
<mails account-name="corpus" folder="INBOX" timestamp="2013-09-08T08:10:18.640Z">
  <mail id="1"/>
</mails>
```

# Usage

censhare

- “html” Widget of XML-Editor
  - General
    - HTML version 3.2
    - No frames
    - No applets
    - No JavaScript

# Usage

censhare

- “html” Widget of XML-Editor
  - “control” parameter
    - Give access to the size of the “html” widget as a parameter to the transformation
    - Result of the transformation can be adapted to the target size
    - Only available, if attribute “include-size-in-control-slot” is set to “true”
    - Example
      - <control width=”5397” height=”492”/>

# Usage

censhare

- “html” Widget of XML-Editor
  - Transformation in dialog definition
    - CSS styles can be defined in “modules/client/common/styles.xml” and referenced with attribute “css-key”
    - Parameters
      - $system, $control
    - Example
      - <xe:html margin=”0” css-key=”asset-info”>
        <xsl:stylesheet xmlns:xsl=”http://www.w3.org/1999/XSL/Transform” version=”2.0”>
        <xsl:param name=”system”/>
        <xsl:template match=”/”>
        <html>
        <xsl:value-of select=”concat('Test with HTML-Widget for asset ', asset/@name)”/>
        </html>
        </xsl:template>
        </xsl:stylesheet>
        </xe:html>

# Usage

censhare

- “html” Widget of XML-Editor
  - Transformation in referenced dialog definition
    - Transformation is stored in “modules/client/common/client-dialog-include.xml” and referenced with attribute “expression-id”
    - CSS styles can be defined in “modules/client/common/styles.xml” and referenced with attribute “css-key”
    - Parameters
      - $system, $control
    - Example
      - <xe:html weight-x=”1.0” weight-y=”1.0” prefwidth=”200” margin=”0” css-key=”asset-info” expression-id=”asset-info”/>

# Usage

censhare

- “html” Widget of XML-Editor
  - Transformation in resource asset
    - Transformation is stored as resource asset and referenced with attribute “url” as a REST URL
    - No CSS style definitions
    - Parameters
      - $transform
    - Example
      - <xe:html weight-x=”1.0” weight-y=”1.0” prefwidth=”200” margin=”0” url=”=:concat($system/system/@censhare-url-prefix, 'assets/asset/id/', asset/@id, '/transform;key=transform:test-parameter;format=html')”/>

# Usage

censhare

- “webbrowser” Widget of XML-Editor
  - XSLT can only be used limited
    - “webbrowser” widget supports “censhare:///” URLs in the “url” attribute
    - “censhare:///” URLs are not supported in the HTML content
    - Therefore, no placed pictures of assets are possible in HTML content

# Usage

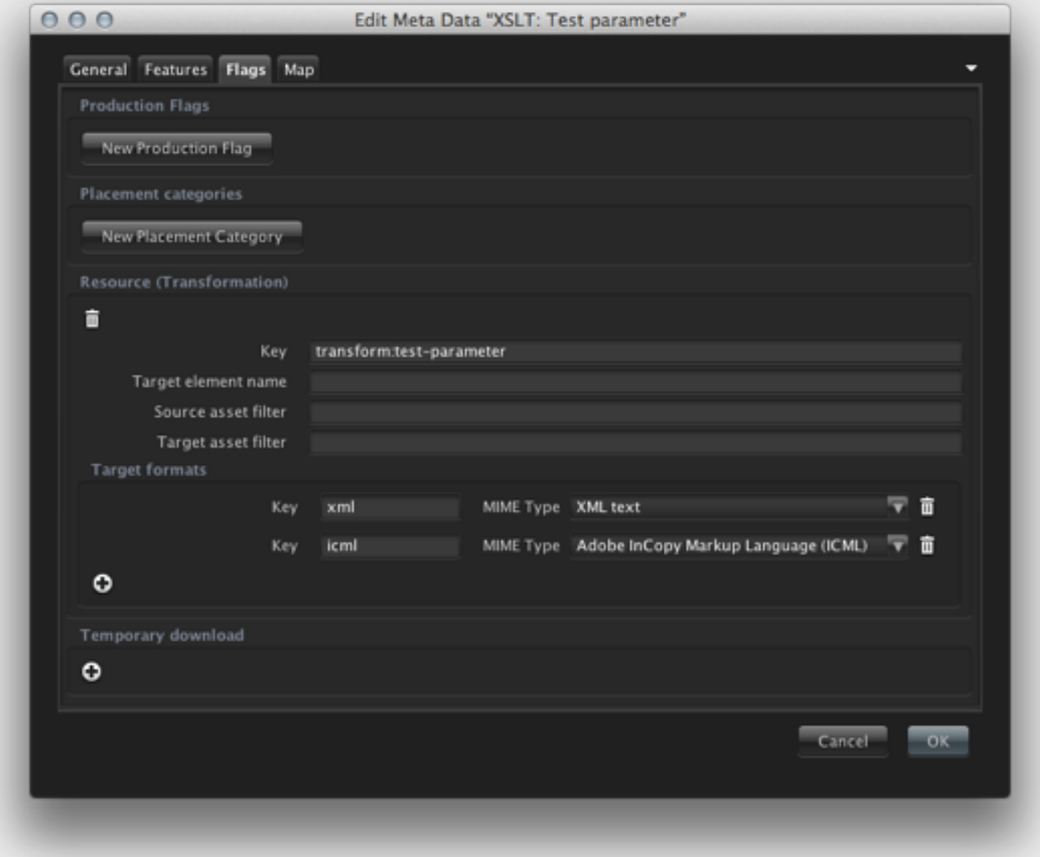
censhare

- Place asset with transformation in Adobe InDesign
  - Target is a ICML (Adobe InCopy Markup Language)
  - Text and anchored boxes with text or placed asset pictures can be created

# Usage

censhare

- Place asset with transformation in Adobe InDesign
  - Transformation must be stored as resource asset with target format “ICML”



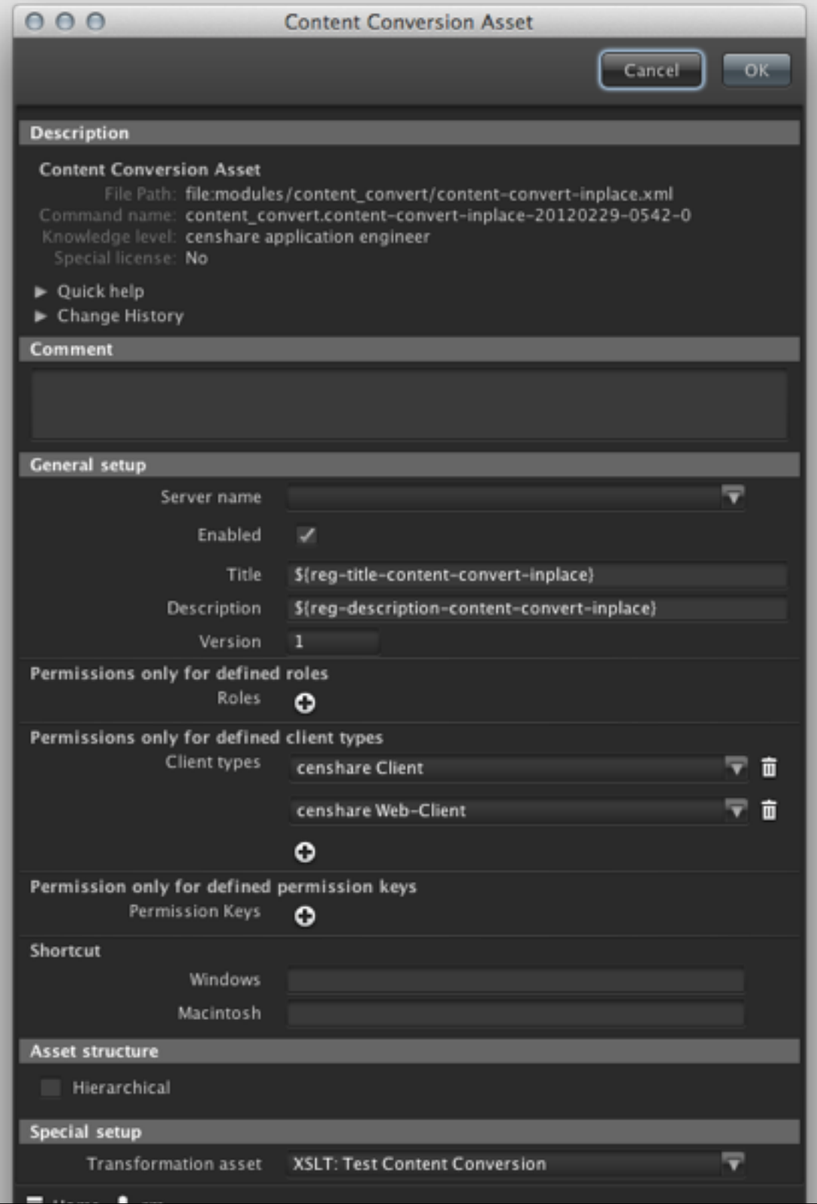
# Usage

censhare

- Place asset with transformation in Adobe InDesign
  - Transform parameter
    - Attributes
      - “mimetype”: Used MIME type
      - “asset-id”: asset ID of placed asset
      - “target-asset-id”: asset ID of target document
      - “format”: target format
      - “key”: transformation key
    - Example
      - <transform mimetype=”application/vnd.adobe.incopy-icml” asset-id=”205371” target-asset-id=”191000” format=”icml” key=”transform:test-parameter”/>

# Usage

- Content conversion
  - New module in censhare 4.4 to convert the XML content of an asset with a transformation asset to a new version of the asset
  - It's available as Server-Action and Asset-Automation
  - Can be used to change the content or the structure of any XML content (XML, ICML, HTML, …) of the master file of assets



# More info

censhare

- Links
  - W3C – XSLT 2.0
    - http://www.w3.org/TR/xslt20/
  - Wikipedia – XSLT
    - http://en.wikipedia.org/wiki/Xslt
  - Zvon.org – XSLT Tutorial in English and German
    - http://zvon.org/xxl/XSLTutorial/Output/index.html
    - http://zvon.org/xxl/XSLTutorial/Output_ger/index.html

# Thank you