Oracle AutoVue

DMAPI Manual

Contents

1 Introduction

AutoVue is a viewing and collaboration solution for enterprise-wide data access. Out of the box, it provides powerful viewing and markup functionalities.

AutoVue's Document Management Application Programming Interface (DMAPI) allows for a tight-knit integration with back-end document management/database systems.

There are several categories of requests that the AutoVue Server makes to the Web server. These include:

1. The initial retrieval of the document from the document repository.

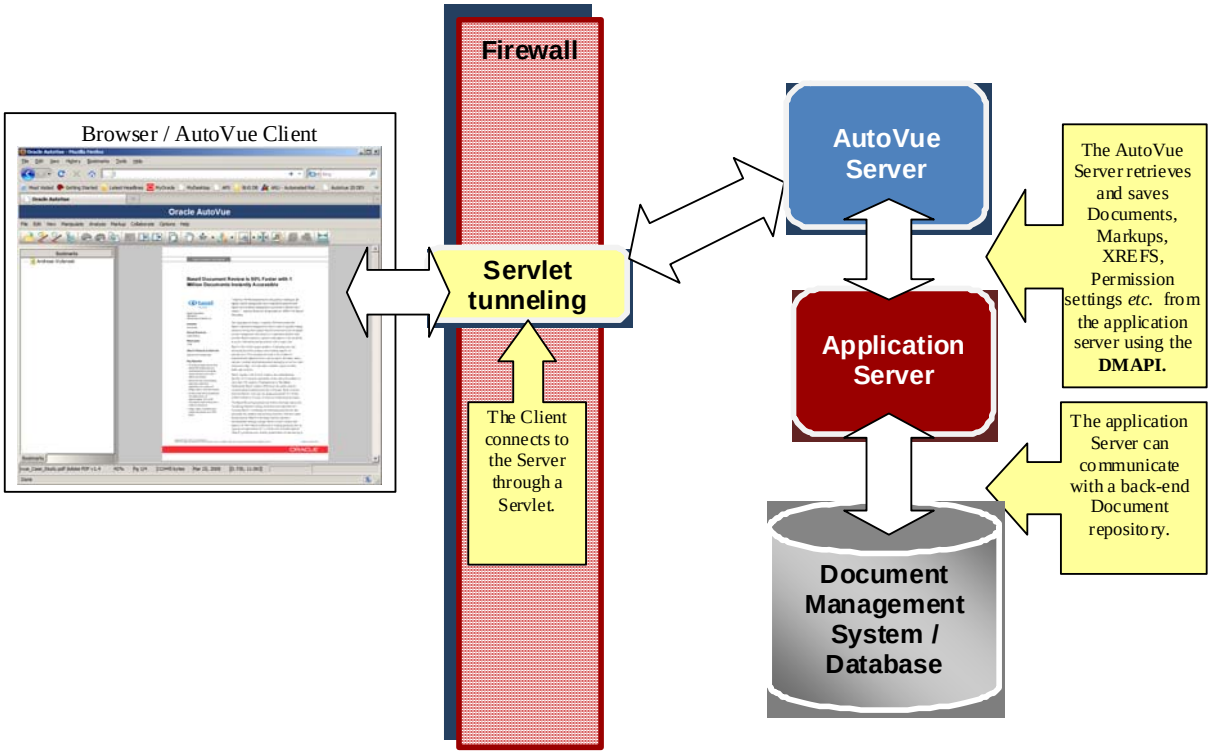
2. Request for properties of the base file. For example:
   - The document title
   - Number of Markups
   - Master markups
   - The list of associated documents such as external reference files (XREFS).

3. Download of associated documents such as XRefs and fonts from the document management system to the AutoVue Server.

4. Markup related requests. For example:
   - Listing markups
   - Opening markups
   - Saving markups

5. Renditions related requests. For example:
   - Saving renditions / cached metafiles

The DMAPI encodes all request parameters and results in the industry standard XML format to provide for future enhancements and simplify integration implementations.



The above diagram illustrates the fundamental architecture of the AutoVue / Document Management environment. In real-world deployments, multiple AutoVue Servers may be present, and the client may be communicating with the servers through a proxy server. These facilities can provide performance and security enhancements but do not affect the implementation of the DMAPI.

Normally, users access AutoVue through a Web browser. An Applet tag in the HTML source of the web page reserves space and specifies the document to be displayed.
For example:

```
<APPLET
 NAME="JVue"
 CODE="com.cimmetry.jvue.JVue.class"
 WIDTH=100% HEIGHT=100% MAYSCRIPT
 ETC.>

<PARAM NAME="FILENAME"
VALUE="http://www.mycompany.com/etc/name?param1=xxx&param2=yyy&etc.">

<PARAM NAME="JVUESERVER"
VALUE="http://www.mycompany.com/servlet/VueServlet">
<PARAM NAME="xxx" VALUE="xxx">
ETC.
</APPLET>
```

Details of the APPLET tag are left out, but the important Applet Parameters that affect the DMAPI are the **FILENAME, JVUESERVER,** and **DMS** tags.

- **FILENAME:** Specifies the URL of the document to view. For a Document Management backend, it is generally a cgi-based URL.

- **JVUESERVER:** Specifies the mechanism(s) to connect to AutoVue Server. The parameter must specify a connection to the AutoVue Server through a Servlet.

- **DMS**: Specifies the URL to the DMAPI server component (servlet, CGI (Common Gateway Interface) script, ASP (Active Server Pages) script, or JSP (Java Server Pages).

The DMAPI integration is a Web server component that is generally implemented as a Servlet. It is called by the AutoVue Server and acts as the intermediary between the AutoVue Server and the back-end Document Management System (DMS). This document describes the format of the communication (request/response) between the AutoVue Server and the DM Integration component.

2 File Open Request

The AutoVue Server will make requests to get the document ID of the file:

```
POST:http://www.server.com/servlet/avdms
```

The POSTed data contains XML data of the following form:

```
<?xml version="1.0"?>
<!DOCTYPE CSIDATA SYSTEM "csi.dtd">   ; Optional

<CSIDATA>
  <Request>
    ; Original FILENAME
    <OriginalURL>
      <![CDATA[filename_parameter]]>
    </OriginalURL>
    <Authorization>
      ; Browser cookie
      <Cookie>
        <![CDATA[Cookie]]>
      </Cookie>
      ; DMS session id
      <Property name="CSI_DMSSession">
       <![CDATA[sessionID]]>
      </Property>
      <Property name="id1"    value="value"/>
      <Property name="id2"    value="value"/>
      ; ETC.
    </Authorization>
    <Open/>
  </Request>
</CSIDATA>
```

Where ***filename_parameter*** is the **FILENAME** PARAM passed in AutoVue applet ("JVue") Tag. It is passed in a CDATA section to allow the use of characters that would otherwise be parsed as XML markup characters ('<', '>', '&', etc).

The response is a XML data containing the document ID property:

```
<?xml version="1.0"?>
<!DOCTYPE CSIDATA SYSTEM "csi.dtd">   ; Optional

<CSIDATA>
  <Response>
    <Property name="CSI_DocID">
      <![CDATA[DocID]]>
    </Property>
```

```
    <ERROR>
      <SUCCEED code="number" message="message" />
    </ERROR>
  </Response>
</CSIDATA>
```

If the DMAPI Servlet is *not* capable of handling '`OriginalURL`' URL (URL does not belong to this DMS), or further authorization is required, then an error message should be returned. Refer to the "**Error/Authorization Handling**" chapter for more information.

3 File Download Request

The AutoVue Server will make requests for several different cases. For example:

1. To download the original file specified in the "**FILENAME**" Applet tag.
2. To download support files such as XRefs.
3. To download markups.
4. To download files for Overlays or File-Comparison.

POST:http://www.server.com/servlet/avdms

The POSTed data contains XML data of the following form:

```
<?xml version="1.0"?>
<!DOCTYPE CSIDATA SYSTEM "csi.dtd">   ; Optional

<CSIDATA>
  <Request>
    ; Original FILENAME (optional)
    <OriginalURL>
      <![CDATA[filename_parameter]]>
    </OriginalURL>
    ; Document ID (mandatory)
    <CSI_DocID>
      <![CDATA[docID]]>
    </CSI_DocID>
    <Authorization>
      ; Browser cookie
      <Cookie>
        <![CDATA[Cookie]]>
      </Cookie>
      ; DMS session id
      <Property name="CSI_DMSSession">
       <![CDATA[sessionID]]>
      </Property>
      <Property name="name" value="value" />
      <Property name="name" value="value" />
      ; ETC.
    </Authorization>
    <Download/>
  </Request>
</CSIDATA>
```

Where ***filename_parameter*** is the **FILENAME** PARAM passed in AutoVue applet ("JVue") Tag. And ***docID*** is the ID of the document to be downloaded.

The data is returned as Content-type **application/octet-stream**.

In the case of an error (such as insufficient Authorization), an < ERROR> **Response** is returned as Content-type text/xml. Refer to the "**Error/Authorization Handling**" chapter for more information.

In a distributed environment where several remote cache servers may be used for storing files, the main DMS server component may redirect the download request to another DMS server component that is installed closer to where files actually reside. This greatly improves performance since AutoVue Server is installed at same location as the remote cache server.

Redirection is done by returning a ticket authorizing AutoVue Server to download the file directly from another location specified in the redirection response:

```
<?xml version="1.0"?>
<CSIDATA>
 <Response>
  <Property name="Redirect">
    ; Redirection type is HTTP URL
    <Property name="Type" value="URL" />
    ; Ticket authorizing download from remote file cache server
    <Property name ="Ticket">"
        "<![CDATA[TicketValue]]>"
    "</Property>
    ; URL to DMS server component located at remote file cache server
    <Property name="Server" value="URL" />
    ; Original FILENAME (optional)
    <Property name ="OriginalURL">"
        <![CDATA[originalURL]]>"
    </Property>
    ; Document ID (mandatory)
    <Property name ="CSI_DocID">
        <![CDATA[docID]]>
    </Property>
  </Property>
 </Response>
</CSIDATA>
```

AutoVue adds the ticket to Authorization block of the Download request made to the DMS server component located at remote cache site.

When the base is a zipped file:

- In case the base file is a zip file, Download request may include '**CSI_BaseDocName**' property.
- DM Integration component can use this property to identify base file for which Download should return metafile (if exists).
- '**CSI_BaseDocName**' property may be nested in case a zip file contains another zip file.
- In AutoVue, format for a URL to a zip file is : `url_to_zip//url_to_another_zip//file_path`

```
<Properties>
   ; Setting base document name for nested zip file
    <Property name="CSI_BaseDocName" >
      <![CDATA[url_to_another_zip]]>
      <Property name="CSI_BaseDocName" >
              <![CDATA[file_path]]>
      <Property/>
    <Property/>
</Properties>
```

4 Getting Properties

The AutoVue Server will make requests for document or collaboration session properties. For example:

1. To get basic document properties such as Title, Author, Creation-Date, Last-Modified-Date etc.
2. To get the list of associated XRefs.
3. To get the list of Markups associated with a document.
4. To get a list of available properties.
5. Retrieving the value of any of the available properties.
6. Getting the user list for a given collaboration session

The request is described below:

POST:http://www.server.com/servlet/avdms

The POSTed data contains XML data of the form:

```
<?xml version="1.0"?>
<!DOCTYPE CSIDATA SYSTEM "csi.dtd">   ; Optional

<CSIDATA>
  <Request>
    ; Original FILENAME (optional)
    <OriginalURL>
      <![CDATA[filename_parameter]]>
    </OriginalURL>
    ; If querying properties for a document
    <CSI_DocID>
      <![CDATA[docID]]>
    </CSI_DocID>
    ; Or if querying properties for a collaboration session
    <CSI_ClbSessionData>
      <![CDATA[dmsClbSessionID]]>
    </CSI_ClbSessionData>
    <Authorization>
      ; Browser cookie
      <Cookie>
        <![CDATA[Cookie]]>
      </Cookie>
      ; DMS session id
      <Property name="CSI_DMSSession">
       <![CDATA[sessionID]]>
      </Property>
      <Property name="name" value="value" />
      ; ETC.
    </Authorization>
    <GetProperties>
```

```
      <Property name="PropertyOne"/>  ; Optional
      <Property name="PropertyTwo" index="number"/> ;index is optional
      ; ETC.
    </GetProperties>
  </Request>
</CSIDATA>
```

Where ***filename_parameter*** is the **FILENAME** PARAM passed in AutoVue applet ("JVue") Tag.

The response is XML structured data:

```
<?xml version="1.0"?>
<!DOCTYPE CSIDATA SYSTEM "csi.dtd">  ; Optional

<CSIDATA>
  <Response>
    <GetProperties>
    ; Public key
       <Property name="PK">
          <![CDATA[publicKeyData]>
     </Property>
    ; Custom data returned as child elements
      ;XREFs
      <Property name="CSI_XREFS">
        <Property name = "XREF">
          <Property name="CSI_DocID">
                <![CDATA[xrefDocID]]>
          </Property>
          <Property name= "Name"  value = "basename"/>
          ; Nested XRef(s)
          <Property name = "XREF">
           <Property name="CSI_DocID">
             <![CDATA[xrefDocID]]>
           </Property>
           <Property name= "Name" value = "xrefname"/>
          </Property>
          ; ETC.
        </Property>
        ; ETC.
      </Property>
      ;Markups
      <Property name="CSI_Markups">
       ;Specify the markup GUI for open and save
        <Property name = "GUI">
          <Property name = "DisplayOptions">
            <Property name="AllowDelete" value="true|false" />
            <Property name="ShowPreviousVersions" value="true|false"/>
            <Property name="AllowNew" value="true|false"/>
            <Property name="AllowImport" value="true|false"/>
            <Property name="AllowExport" value="true|false"/>
            <Property name="AllowNewLayers" value="true|false"/>
```

```
            <Property name="AllowModifyLayers" value="true|false"/>
          </Property>
          <Property name = "Display">
            <Property name="name1" value="width" />
            <Property name="name2" value="width" />
          </Property>
          <Property name = "Edit">
            <STATIC value="value">
            <EDIT   id="name1" name="title1" hidden ="false|true" />
            <COMBO  id="name2" name="title2" default="default">
              <Value value="value1"/>
              <Value value="value1"/>
              <Value value="value1"/>
            </COMBO>
            ; ETC.
          </Property>
        </Property>



        <Property name = "Markup">
          <Property name="CSI_DocID">
               <![CDATA[markupDocID]]>
          </Property>
          ; for read only markups
          <Property name= "CSI_DocReadOnly" value="true|false"/>
          ; ETC.
          <Property name= "name1" value="value1"/>
          <Property name= "name2" value="value2"/>
          ; ETC.
        </Property>
        <Property name = "Markup">
          <Property name="CSI_DocID"
              <![CDATA[markupDocID]]>
          </Property>
          <Property name= "name1" value="value1"/>
          <Property name= "name2" value="value2"/>
          ; ETC.
        </Property>
        ; ETC.
      </Property>

       ;Versions
       <Property name="CSI_Versions">
          <Property name="CSI_DocID">
                  <![CDATA[versionDocID]]>
          </Property>
          ; ETC.
       </Property>
       ;Collaboration
       <Property name="CSI_Collaboration">
```

```
<Property name = “GUI”>
 ;Specify the session creation options
  <Property name = “DisplayOptions”>
    <Property name=“AllowAdd” value=”true|false” />
    <Property name=“AllowAddNew” value=”true|false”/>
    <Property name=“AllowRemove” value=”true|false”/>
    <Property name=“AllowLayerColor” value=”true|false”/>
  </Property>
  ;Specify the session information fields
  <Property name = “Display”>
    <Property name=”name1” value=”width” />
    <Property name=”name2” value=”width” />
  </Property>
  ;Specify the session invitation elements
  <Property name = “Invitation”>
    <STATIC value=”value”>
    <EDIT   id=”name1” name=”title1” hidden =”false|true” />
    <COMBO  id=”name2” name=”title2” default=”default”>
      <Value value=”value1”/>
      <Value value=”value1”/>
      <Value value=”value1”/>
    </COMBO>
    <LIST  id=”name2” name=”title2”>
      <Value value=”value1”/>
      <Value value=”value1”/>
      <Value value=”value1”/>
    </LIST>
    ; ETC.
  </Property>
</Property>
<Property name = “Session”>
  <Property name="CSI_ClbSessionID">
       <![CDATA[collaborationSessionID]]>
  </Property>
  ; for private/public session, default public
 <Property name= ”CSI_ClbSessionType” value=”private|public”/>
  ; Save chat transcript at end of session
  <Property name= ”CSI_ClbSaveChat” value=”true|false”/>
  ; ETC.
  <Property name= ”name1” value=”value1”/>
  <Property name= ”name2” value=”value2”/>
   ; ETC.
</Property>
<Property name = “Session”>
  <Property name="CSI_SessionID">
       <![CDATA[collaborationSessionID]]>
 </Property>
  <Property name= ”name1” value=”value1”/>
  <Property name= ”name2” value=”value2”/>
   ; ETC.
</Property>
```

```
        ; ETC.
      </Property>
      ;Permissions
      <Property name="CSI_Permissions">
        <Property name = "GUI">
          <Property name ="Disable"/>   ; Optional
          <Value value="id1" />
          <Value value="id2" />
          ; ETC.
        </Property>
      </Property>
      ; Single valued property
      <Property name="PropertyOne" value="Value" />
      ; Multi valued property
      <Property name="PropertyTwo">
        <Value value="Value1" />
        <Value value="Value2" />
        <Value value="Value3" />
      </Property>
      ; Renditions
      <Property name = "CSI_Renditions" value="Value">
        ; Optional, if Metafile exist
        <Property name="CSI_DocID">
          <![CDATA[MetafileRenditionDocID]]>
        </Property>
      </Property>
      ; ETC.
      ; List of All Properties
      <Property name = "CSI_ListAllProperties" >
        <Property name="Property1" />
        <Property name="Property2" />
        <Property name="Property3" />
        ; ETC.
      </Property>
      ; DMS session id
      <Property name="CSI_DMSSession">
        <![CDATA[sessionID]]>
      </Property>
      ; GUI for Login
      <Property name = "GUI" value = "Authorization">
        <Property name = "Edit">
          <STATIC value="value">
          <EDIT   id="name1" name="title1" hidden ="false|true" />
          <COMBO  id="name2" name="title2" default="default">
            <Value value="value1"/>
            <Value value="value2"/>
            <Value value="value3"/>
          </COMBO>
          ; ETC.
        </Property>
      </Property>
```

```
      ; GUI for File Browse
      <Property name = “GUI” value = “Browse”>
        <Property name = “Display”>
          <Property name=”name1” value=”width” />
          <Property name=”name2” value=”width” />
         ; ETC.
        </Property>
      </Property>
      ; Specify List of items that are direct children
      <Property name=”CSI_ListItems”>
        <Property name="CSI_DocID">
          <![CDATA[docID]]>
          ; Item type : folder or document
          <Property name= ”CSI_ItemType”
value=”CSI_Folder|CSI_Document”/>
          ; Other properties
          <Property name= ”name1” value=”value1”/>
          <Property name= ”name2” value=”value2”/>
         ; ETC.
         ; Nested doc id imply children (optional)
         <Property name="CSI_DocID">
          <![CDATA[docID]]>
          ; Item type : folder or document
          <Property name= ”CSI_ItemType”
value=”CSI_Folder|CSI_Document”/>
          ; Other properties
          <Property name= ”name1” value=”value1”/>
          <Property name= ”name2” value=”value2”/>
          ; ETC.
         </Property>
        </Property>
      ; GUI for Search
      <Property name=”GUI” value =”Search”/>
        ; Specify GUI for search form
        <Property name = “Edit”>
          <STATIC value=”value”>
          <EDIT   id=”name1” name=”title1” hidden =”false|true” />
          <COMBO  id=”name2” name=”title2” default=”default”>
            <Value value=”value1”/>
            <Value value=”value2”/>
            <Value value=”value3”/>
          </COMBO>
          ; ETC.
        </Property>
        ; Specify GUI for search results
        <Property name = “Display”>
          <Property name=”name1” value=”width” />
          <Property name=”name2” value=”width” />
        </Property>
      </Property>
      ; Search results
```

```
      <Property name="CSI_Search">
        ; Specify the List of items that match search criteria
        <Property name="CSI_DocID">
          <![CDATA[docID]]>
          ; Other properties
          <Property name= "name1" value="value1"/>
          <Property name= "name2" value="value2"/>
          ; ETC.
        </Property>
       ; ETC.
      </Property>
      ; GUI for Document Check-in
      <Property name = "GUI" value = "Document">
        ; Allow Browse option
        <Property name = "DisplayOptions">
           <Property name="AllowBrowse" value="true|false" />
        </Property>
        <Property name = "Edit">
          <STATIC value="value">
          <EDIT   id="name1" name="title1" hidden ="false|true" />
          <COMBO  id="name2" name="title2" default="default">
            <Value value="value1"/>
            <Value value="value2"/>
            <Value value="value3"/>
          </COMBO>
          ; ETC.
        </Property>
      </Property>
      ; CAD/DMS Attributes Mapping
      <Property name="CSI_CADAttributesMapping">
         <Property name= "dmsName1" value="cadName1"
readOnly="true|false" />
         <Property name= "dmsName2" value="cadName2"
readOnly="true|false" />
         ; ETC.
      </Property>

; Markup Policy
     <Property name="CSI_MarkupPolicy">
          <![CDATA[
               XML content of markup policy
          ]]>
     </Property>

     ; IntelliStamps
     <Property name="CSI_IntelliStamp">
          <Property name="CSI_IntelliStamp_Definition">
               <![CDATA[
                    content of dmstamps.ini file
               ]]>
          </Property>
```

```
            <Property name="CSI_IntelliStamp_Images">
                  <Property name="CSI_IntelliStamp_Image">
                        <Property name="CSI_DocName">
                              <![CDATA[stamp_image_name_1]]>
                        </Property>
                        <Property name="CSI_DocID">
                              <![CDATA[stamp_docid_1]]>
                        </Property>
                  </Property>

                  <Property name="CSI_IntelliStamp_Image">
                        <Property name="CSI_DocName">
                              <![CDATA[stamp_image_name 2]]>
                        </Property>
                        <Property name="CSI_DocID">
                              <![CDATA[stamp_docid_2]]>
                        </Property>
                  </Property>

                  ; ETC. - Other images

            </Property>
      </Property>

      ; ETC.
       </GetProperties>
  </Response>
</CSIDATA>
```

Note:

- The request for Properties is flexible. Any number of properties can be requested simultaneously, or the requests can be separated.

- The Property `CSI_Versions` is predefined. The returned data is a list of previous version doc ids:

```
<Property name = “CSI_Versions”>
  <Property name="CSI_DocID">
    <![CDATA[versionDocID]]>
  </Property>
</Property>
```

The Property `VERSION` is predefined. The returned data can contain additional information such as doc names and version numbers in addition to ids. A series of zeros or more `VERSION` entries with properties `CSI_DocID`, `CSI_DocName` and `CSI_Version` may be returned.

```
<Property name = “CSI_Versions”>
  <Property name="VERSION">
```

```
    <Property name="CSI_DocID">
      <![CDATA[versionDocID]]>
    </Property>
    <Property name="CSI_DocName" value="DocName"/> ; mandatory
    <Property name="CSI_Version" value="version"/> ; mandatory
  </Property>
</Property>
```

- The Property **CSI_Renditions** is predefined. The returned data should conform to the following format:

```
; for single valued
<Property name = "CSI_Renditions" value="Value">
; optional, if Metafile exist
  <Property name="CSI_DocID">
    <![CDATA[MetafileRenditionDocID]]>
  </Property>
</Property>

; for multi valued
<Property name = "CSI_Renditions" >
  <Value value="Value1" />
  <Value value="Value2" />
  <Value value="Value3" />
  … Etc …
  ; optional, if Metafile exist
  <Property name="CSI_DocID">
    <![CDATA[MetafileRenditionDocID]]>
  </Property>
</Property>
```

The value for CSI_Renditions is a list of allowed renditions. If Metafile rendition is supported, CSI_META should be included in the list. If a Metafile rendition exists, its doc id should also be returned with property name CSI_DocID. If CSI_DocID were returned, AutoVue Server would issue a request to download the Metafile.

- The Property CSI_ListAllProperties is predefined. The returned data should conform to the following format:

```
<Property name = "CSI_ListAllProperties" >
  <Property name="Property1" />
  <Property name="Property2" />
  <Property name="Property3" />
  … Etc …
</Property>
```

The returned list of properties is used by AutoVue to fill in the print codes for Footers/Headers/Watermark in Print Properties dialog box and the File / Properties / DMS dialog box.

- The Property CSI_XREFS is predefined. The returned data should conform to the following format:
  A series of zeros or more XRef entries with properties name and CSI_DocID:

```
<Property name = "XREF">
  <Property name="CSI_DocID">
      <![CDATA[xrefDocID]]>
  </Property>
  <Property name= "Name" value = "xrefname"/>
</Property>
```

The Name property corresponds to the external name of the document and the DocID corresponds to the unique Document ID of that identifies the Document in the DMS. Both Name and CSI_DocID are required properties.

Nested XREFs:

- List of XRefs may include nested XRefs to support Assembly/Parts and Master/XRefs designs:

```
<Property name = "XREF">
  <Property name="CSI_DocID">
      <![CDATA[xrefDocID]]>
  </Property>
  <Property name= "Name" value = "xrefname"/>
  ; Nested XRef(s)
  <Property name = "XREF">
    <Property name="CSI_DocID">
      <![CDATA[xrefDocID]]>
    </Property>
    <Property name= "Name" value = "xrefname"/>
  </Property>
  ; ETC.
</Property>
 ; ETC
```

- The Property CSI_Markups is predefined. The returned data should conform with the following format:
  A property, named GUI, which contains a list of GUI element definitions. The Display attribute lists the attributes to be displayed in the Markup selection dialog box along with the width to reserve for the attribute's display. The display width is specified as a number of characters.
  The Edit property allows the server to define the user interface presented to the user when a markup save is requested. The children of edit may include user interface elements that define static text (STATIC), edit boxes (EDIT), and combo boxes with a list of possible choices (COMBO). For each user interface element, "id" identifies the name of the attribute that the server will pass back to the integration, and "name" specifies the caption for the element. Combo box options are specified as nested "Value" tags.

After the user interface definitions, zero or more `Markup` properties follow. Each entry contains one or more properties that can be displayed in the Markup selection dialog box. The attribute CSI_DocID is required for each Markup property.

```
<Property name="CSI_Markups">
 ;Specify the markup GUI for open and save
  <Property name = "GUI">
    <Property name = "DisplayOptions">
      <Property name="AllowDelete" value="true|false" />
      <Property name="ShowPreviousVersions" value="true|false"/>
      <Property name="AllowNew" value="true|false"/>
      <Property name="AllowImport" value="true|false"/>
      <Property name="AllowExport" value="true|false"/>
      <Property name="AllowNewLayers" value="true|false"/>
      <Property name="AllowModifyLayers" value="true|false"/>
    </Property>
    <Property name = "Display">
      <Property name="name1" value="width" />
      <Property name="name2" value="width" />
     ; ETC.
    </Property>
    <Property name = "Edit">
      <STATIC value="value"/>
      <EDIT   id="name1" name="title1" hidden ="false|true" />
      <COMBO  id="name2" name="title2" default="default">
        <Value value="value1"/>
        <Value value="value1"/>
        <Value value="value1"/>
      </COMBO>
      ; ETC.
    </Property>
  </Property>
   <Property name = "Markup">
     <Property name="CSI_DocID">
        <![CDATA[markupDocID]]>
     </Property>
     <Property name= "name1" value="value1"/>
     <Property name= "name2" value="value2"/>
        ; ETC.
   </Property>
   <Property name = "Markup">
     <Property name="CSI_DocID">
        <![CDATA[markupDocID]]>
     </Property>
     <Property name= "name1" value= "value1"/>
     <Property name= "name2" value= "value2"/>
     ; ETC.
    </Property>
    ; ETC.
  </Property>
```

Collaboration:

- If AutoVue applet is launched with COLLABORATION parameter with VALUE=**"INIT"**, then AutoVue server queries the DMS server-side component for collaboration session id by calling `GetProperties(`**`CSI_ClbSessionID`**`)`. The DMS server-side component can extract the session id from this session data. The session id should be unique.

- The Property CSI_Collaboration is predefined. The returned data should conform with the following format:
  - A property, named "**GUI"**, which contains a list of GUI element definitions.
  - **"DisplayOptions"** section for enabling/disabling GUI items in Invitation dialog box.
  - The "**Display**" attribute lists the attributes to be displayed in the Session selection dialog along with the width to reserve for the attributes display. The display width is specified as a number of characters.
  - The "**Invitation**" property allows the server to define the user interface presented to the user when an invitation entry is added. It may include user interface elements that define static text (STATIC), edit boxes (EDIT), and combo boxes with a list of possible choices (COMBO). For each user interface element, "id" identifies the name of the attribute that the server will pass back to the integration, and "name" specifies the caption for the element. Combo box options are specified as nested "Value" tags.
  - A **`CSI_ClbUsers`** section lists users who have already been invited to the collaboration.
  - A **`Session`** section lists session information (properties) such as session title, id, type, subject, duration, start time…etc.
  - **`CSI_ClbSaveChat`** indicates whether DMS server-side component supports saving chat transcript.

  After the user interface definitions, zero or more Session properties follow. Each entry contains one or more properties that can be displayed in the Session selection dialog box. These session entries relate to the DMS-defined collaboration entries and not to the actual AutoVue Server collaboration entries.

```
<Property name="CSI_Collaboration">
  <Property name = "GUI">
   ;Specify the session creation options
    <Property name = "DisplayOptions">
      <Property name="AllowAdd" value="true|false" />
      <Property name="AllowAddNew" value="true|false"/>
      <Property name="AllowRemove" value="true|false"/>
      <Property name="AllowLayerColor" value="true|false"/>
    </Property>
    ;Specify the session information fields
    <Property name = "Display">
      <Property name="name1" value="width" />
      <Property name="name2" value="width" />
    </Property>
    ;Specify the session invitation elements
    <Property name = "Invitation">
      <STATIC value="value">
```

```
            <EDIT   id="name1" name="title1" hidden ="false|true" />
            <COMBO  id="name2" name="title2" default="default">
              <Value value="value1"/>
              <Value value="value1"/>
              <Value value="value1"/>
            </COMBO>
            ; ETC.
          </Property>
        </Property>
        <Property name = "ClbUsers">
           <Value value ="user1"/>
           <Value value ="user2"/>
        </Property>
        <Property name = "Session">
          <Property name="CSI_ClbSessionID">
               <![CDATA[collaborationSessionID]]>
          </Property>
          ; for private/public session, default public
         <Property name= "CSI_ClbSessionType" value="private|public"/>
          ; Save chat transcript at end of session
          <Property name= "CSI_ClbSaveChat" value="true|false"/>
          ; ETC.
          <Property name= "name1" value="value1"/>
          <Property name= "name2" value="value2"/>
           ; ETC.
        </Property>
        <Property name = "Session">
          <Property name="CSI_SessionID">
               <![CDATA[collaborationSessionID]]>
         </Property>
          <Property name= "name1" value="value1"/>
          <Property name= "name2" value="value2"/>
           ; ETC.
        </Property>
        ; ETC.
      </Property>
```

DMS Session:

- The property **CSI_DMSSession** is predefined. Authorization properties must include login information such as user name (**CSI_UserName**), password (**CSI_Password**), domain (**CSI_Domain**) and library (**CSI_Library**):

```
<Authorization>
  ; Login info
  <Property name="CSI_UserName"  value="value"/>
  <Property name="CSI_Password"  value="value"/>
  <Property name="CSI_Domain"    value="value"/>
  <Property name="CSI_Library"   value="value"/>
  <Property name="id"             value="value"/>
  ; ETC.
</Authorization>
<GetProperties>
  ; DMS Session id
  <Property name="CSI_DMSSession">
</GetProperties>
```

- The returned data should include **DMS Session ID** and conform with the following format:

```
; DMS session id
<Property name="CSI_DMSSession">
  <![CDATA[sessionID]]>
</Property>
```

- The properties '**GUI**' and '**Authorization**' are predefined:

```
<GetProperties>
  ; GUI for Login dialog
  <Property name="GUI" value ="Authorization"/>
</GetProperties>
```

- The returned data should include **GUI** items such as CSI_UserName, CSI_Password, CSI_Domain and CSI_Library which describe the login fields and conform to the following format:

```
; Login GUI
<Property name = "GUI" value = "Authorization">
  <Property name = "Edit">
    <STATIC value="value">
    <EDIT   id="name1" name="title1" hidden ="false|true" />
    <COMBO  id="name2" name="title2" default="default">
      <Value value="value1"/>
      <Value value="value2"/>
      <Value value="value3"/>
    </COMBO>
    ; ETC.
```

```
        </Property>
      </Property>
```

- The properties '**GUI**' and '**Browse**' are predefined:

```
    <GetProperties>
      ; GUI for Browse dialog
      <Property name="GUI" value ="Browse"/>
    </GetProperties>
```

- The returned data should include **GUI** elements that describe the columns for listing the items and conform with the following format:

```
      ; Specify the List GUI
      <Property name = "GUI" value = "Browse">
        <Property name = "Display">
          <Property name="name1" value="width" />
          <Property name="name2" value="width" />
         ; ETC.
        </Property>
      </Property>
```

- The property **CSI_ListItems** is predefined. The XML request may include ***docID*** for which direct children list is to be returned. If ***docID*** is null, the response should include list of items found at the ***root level*** in the DMS:

```
    ; Folder ID for which direct children list is to be returned
    <CSI_DocID>
      <![CDATA[folderID]]>
    </CSI_DocID>
    <GetProperties>
      ; Browse Items
      <Property name="CSI_ListItems" />
    </GetProperties>
```

- The returned data should include list of **items** that are direct children of the doc id passed in the XML request. Each **item** must include the ***docID***:

```
      <Property name="CSI_ListItems">
        ; Specify List of items that are direct children
        <Property name="CSI_DocID">
          <![CDATA[docID]]>
          ; Item type : folder or document
          <Property name= "CSI_ItemType"
value="CSI_Folder|CSI_Document"/>
          ; Other properties
          <Property name= "name1" value="value1"/>
          <Property name= "name2" value="value2"/>
         ; ETC.
         ; Nested doc id imply children (optional)
         <Property name="CSI_DocID">
```

```
          <![CDATA[docID]]>
          ; Item type : folder or document
          <Property name= "CSI_ItemType"
value="CSI_Folder|CSI_Document"/>
          ; Other properties
          <Property name= "name1" value="value1"/>
          <Property name= "name2" value="value2"/>
          ; ETC.
         </Property>
        </Property>
```

DMS Browse:

- The properties '**GUI**' and '**Document**' are predefined:

```
    <GetProperties>
      ; GUI for File checkin dialog
      <Property name="GUI" value ="Document"/>
    </GetProperties>
```

- The returned data should include **GUI** elements that compose the check-in dialog (GUI → Edit). The response may also include '**AllowBrowse**' option to show/hide '**Browse**' button for check-in dialog:

```
      ; GUI for Document Check-in
      <Property name = "GUI" value = "Document">
        ; Allow Browse option
        <Property name = "DisplayOptions">
           <Property name="AllowBrowse" value="true|false" />
        </Property>
        <Property name = "Edit">
          <STATIC value="value">
          <EDIT   id="name1" name="title1" hidden ="false|true" />
          <COMBO  id="name2" name="title2" default="default">
            <Value value="value1"/>
            <Value value="value2"/>
            <Value value="value3"/>
          </COMBO>
          ; ETC.
        </Property>
      </Property>
```

- The property **CSI_CADAttributesMapping** is predefined. The response is an XML data containing a list of DMS attributes mapped to CAD attributes. '**readOnly**' flag indicates whether the DMS attribute can be updated (default is **false**):

```
      <Property name="CSI_CADAttributesMapping">
         <Property name= "dmsName1" value="cadName1"
readOnly="true|false" />
         <Property name= "dmsName2" value="cadName2"
readOnly="true|false" />
```

```
        ; ETC.
    </Property>
```

- The property **CSI_AllowBrowse** is predefined. The response should return 'true' if DMS Server Component supports file browse feature:

```
    <Property name="CSI_AllowBrowse" value="true|false" />
```

- The following Properties are predefined, ones that are related to DMS document attributes are expected to map, internally to the appropriate DMS property:

Property Name	Description
CSI_DocID	Read-only: A unique identifier for the document. This is a read-only attribute.
CSI_BaseDocID	Read-only: A unique identifier for the parent document.
CSI_BaseDocName	File name and path for the parent document
CSI_IsDMSDoc	Read-only: Returns 1 if the specified document/URL belongs to the DMS repository. Returns 0 if it does not (e.g. for a local file, or a URL on a WEB server that is not managed by the DMS).
CSI_IsMultiContent	Read-only: Indicates that the document ID refers to a collection of documents indexed by page number.
CSI_DocName	The descriptive title of the document
CSI_DocSize	Read-only: Size of document in bytes
CSI_DocDateCreate	Read-only: Date document was created (GMT Time)
CSI_DocDateLastModified	Read-only: Date document was last modified (GMT Time)
CSI_DocAuthor	Document author/owner
CSI_UserName	User Name
CSI_Version	The version number (a[,b[,c[,d]]])
CSI_MarkupType	Can be one of normal, master, consolidated. This applies only to Markup documents. The return value should be empty for non-Markup documents.
CSI_MultiContentPage	Index to content when dealing with a multi-content document.
CSI_ListAllProperties	A list of all document attributes available.
CSI_Renditions	A list of allowed renditions format types. Available values are: PCRS_TIF CSI_META
CSI_Versions	A list of docIDs corresponding to all versions of a document.
CSI_RenditionType	Can be one of supported formats returned by CSI_Renditions or “CSI_META” which is used for cached meta files. This applies only to Rendition documents.
CSI_Compression	Used in the Save request to indicate that the document is gzip-compressed. Can be “true” or “false” (default “false”).
CSI_DocReadOnly	Used to specify read only markups, mainly for markups. Can be “true” or “false”, if not present “false” is assumed.
CSI_MIMETypes	Used in the Save request to indicate which

Property Name	Description
	MIME types are supported when uploading data. Can be “text/xml” or “multipart/form-data” (default is “text/xml”).
CSI_Collaboration	Used for collaboration requests
CSI_ClbSessionData	A DMS collaboration session handle
CSI_ClbSessionID	A unique identifier for a collaboration session
CSI_ClbSessionType	Can be either private or public. This applies only to collaboration sessions. The return value should be empty otherwise.
CSI_ClbSaveChat	Saves the session chat transcript and on collaboration session close. Can be true or false, default is false.
CSI_ClbDocType	Used to save chat transcript attached to a session. Only “chat” value is supported.
CSI_ClbUsers	Returns the list of collaboration users. If a session is defined in the request, it requests the list of collaboration users in the session.
CSI_DMSSession	DMS Session id after successful login to DMS.
CSI_Password	Valid password to login to DMS.
CSI_Domain	Domain name to login to DMS.
CSI_Library	Library name to login to DMS.
GUI Authorization Browse Search Document	Used to return user interface items. Login dialog box File/Folder browse dialog box Search form dialog box File Save (check-in) dialog box
CSI_ListItems	Used in file browse to return list of direct children for a give folder id.
CSI_ItemType	Used to indicate item type when returning list of items for CSI_ListItems. Can be either CSI_Folder or CSI_Document.
CSI_Search	Used to perform search for documents inside DMS.
CSI_CADAttributesMapping	Used to return CAD/DMS attributes mapping.
CSI_AllowBrowse	‘true’ if file browse is supported. Otherwise, ‘false’.
CSI_AllowSearch	‘true’ if file search is supported. Otherwise, ‘false’.
CSI_Redirected	‘true’ if Save request should be redirected to another DMS Server Component. Otherwise, ‘false’.
CSI_IntelliStamp	Used to return information about intellistamps supported by DMS Server Component.
CSI_IntelliStamp_Definition	Used to return intellistamp definition file (dmstamps.ini)
CSI_IntelliStamp_Images	Used to return information about underlying images referenced by intellistamps
CSI_IntelliStamp_Image	Used to return information about underlying

Property Name	Description
	image for a specific intellistamp
CSI_MarkupPolicy	Used to return information about markup policy supported by DMS Server Component.
PK	Used to return public key for encrypting Authorization block by AutoVue Server

Property values that are not predefined may be fetched by specifying their name as the value of the property entity in the GetProperty request.

In the case of an error (such as insufficient Authorization), an < ERROR> **Response** is returned. Refer to the "**Error/Authorization Handling**" chapter for more information.

DMS Search:

- The properties '**GUI**' and '**Search**' are predefined:

```
<GetProperties>
  ; GUI for Search dialog
  <Property name="GUI" value ="Search"/>
</GetProperties>
```

- The returned data should include **GUI** elements that describe the columns for search results (GUI → Display) as well as GUI elements that compose the search form (GUI → Edit):

```
; Specify the GUI for search form
<Property name="GUI" value ="Search"/>
  <Property name = "Edit">
    <STATIC value="value">
    <EDIT   id="name1" name="title1" hidden ="false|true" />
    <COMBO  id="name2" name="title2" default="default">
      <Value value="value1"/>
      <Value value="value2"/>
      <Value value="value3"/>
    </COMBO>
    ; ETC.
  </Property>
  ; Specify the GUI for search results
  <Property name = "Display">
    <Property name="name1" value="width" />
    <Property name="name2" value="width" />
  </Property>
</Property>
```

- To perform search, request should include user entered values to properties defined in the search form as follows:

```
<GetProperties>
  <Property name="CSI_Search">
    ; Properties in search form with user entered values
    <Property name= "name1" value="value1"/>
    <Property name= "name2" value="value2"/>
  </Property>
</GetProperties>
```

- The returned data should include list of **items** that match the search criteria. Each **item** must include the ***docID***.

```
; Specify the List of items that match search criteria
<Property name="CSI_Search">
  <Property name="CSI_DocID">
    <![CDATA[docID]]>
    ; Other properties
```

```
          <Property name= "name1" value="value1"/>
          <Property name= "name2" value="value2"/>
          ; ETC.
        </Property>
       ; ETC.
      </Property>
```

- The property **CSI_AllowSearch** is predefined. The response should return 'true' if DMS Server Component supports file search feature:

```
      <Property name="CSI_AllowSearch" value="true|false" />
```

- The property **CSI_BaseDocName** is predefined. In case base file is a zip file, GetProperties request may include '**CSI_BaseDocName**' property.
- DM Integration component can use this property to identify base file for which GetProperties should return. For example, in case of markups, this property can be used to identify which markups belong to which file inside the zip file.
- '**CSI_BaseDocName**' property may be nested in case a zip file contains another zip file.
- In AutoVue, format for a URL to a zip file is :

```
url_to_zip//url_to_another_zip//file_path
```

```
<Properties>
   <Property name="CSI_BaseDocID">
     <![CDATA[basedocID]]>
   </Property>
   ; Setting base document name for nested zip file
    <Property name="CSI_BaseDocName" >
      <![CDATA[url_to_another_zip]]>
      <Property name="CSI_BaseDocName" >
              <![CDATA[file_path]]>
      <Property/>
    <Property/>
</Properties>
```

Note:

CSI_Search and Browse do not require a context and as such can be issued without CSI_OriginalURL, CSI_DocID or CSI_SessionID.

AutoVue Mobile:

- Get Properties request may include '**CSI_IntelliStamp**' property.

```
<GetProperties>
   ; Request for IntelliStamp Definition file
   <Property name="CSI_IntelliStamp"/>
</GetProperties>
```

- The returned data should include
  1. **The definition file for IntelliStamps**
     - This is basically the content of dmstamps.ini file which is generated by 'stampdlg.exe' tool shipped with AutoVue.
  2. **The underplaying images for IntelliStamps**
     - This is basically the name and DocID of each of the underlying images for each intellistamp.
     - AutoVue Server downloads each of the underlying imags by invoking the normal file download request and passing the DocID of the stamp image.

```
<GetProperties>
      <Property name="CSI_IntelliStamp">
            <Property name="CSI_IntelliStamp_Definition">
                  <![CDATA[
                        content of dmstamps.ini file
                  ]]>
            </Property>
            <Property name="CSI_IntelliStamp_Images">
                  <Property name="CSI_IntelliStamp_Image">
                        <Property name="CSI_DocName">
                              <![CDATA[stamp_image_1]]>
                        </Property>
                        <Property name="CSI_DocID">
                              <![CDATA[stamp_docid_1]]>
                        </Property>
                  </Property>

                  <Property name="CSI_IntelliStamp_Image">
                        <Property name="CSI_DocName">
                              <![CDATA[stamp_image_2]]>
                        </Property>
                        <Property name="CSI_DocID">
                              <![CDATA[stamp_docid_2]]>
                        </Property>
                  </Property>

                  ; ETC. - Other images

            </Property>
      </Property>
</GetProperties>
```

- Get Properties response may include '**PickList**' property.
  - This property indicates type of attributes defined inside IntelliStamps.
  - There are 4 different types for PickList:

1. Single Valued & Constrained

- Only one value can be assigned to this type of property
- Valid value is restricted to the pick list. Any value that is not in the pick list would be considered an invalid value.

```
<Property name="name" value="value">
     <Property constrained="true" name="PickList">
          <Property name="PickValue" value="value_1"></Property>
          <Property name="PickValue" value="value_2"></Property>
          <Property name="PickValue" value="value_3"></Property>
     </Property>
</Property>
```

2. Single Valued & Non-Constrained

- Only one value can be assigned to this type of property
- Valid value is not restricted to the pick list. Any value would be considered as valid including user-defined one.

```
<Property name="name" value="value">
     <Property constrained="false" name="PickList">
          <Property name="PickValue" value="value_1"></Property>
          <Property name="PickValue" value="value_2"></Property>
          <Property name="PickValue" value="value_3"></Property>
     </Property>
</Property>
```

3. Multi Valued & Constrained

- Multiple values can be assigned to this type of property
- Valid values are restricted to the pick list. Any value that is not in the pick list would be considered an invalid value.

```
<Property multi="true" name="name">
     <Value value="value_1"></Value>
     <Value value="value_2"></Value>
     <Property constrained="true" multi="true" name="PickList">
          <Property name="PickValue" value="value_1"></Property>
          <Property name="PickValue" value="value_2"></Property>
          <Property name="PickValue" value="value_3"></Property>
     </Property>
</Property>
```

4. Multi Valued & Non-Constrained

- Multiple values can be assigned to this type of property
- Valid values are not restricted to the pick list. Any value would be considered as valid including user-defined ones.

```
<Property multi="true" name="name">
      <Value value="value_1"></Value>
      <Value value="value_2"></Value>
      <Property constrained="false" multi="true" name="PickList">
            <Property name="PickValue" value="value_1"></Property>
            <Property name="PickValue" value="value_2"></Property>
            <Property name="PickValue" value="value_3"></Property>
      </Property>
</Property>
```

o Get Properties request may include '**CSI_MarkupPolicy**' property.

```
<GetProperties>
   ; Request for Markup Policy
   <Property name="CSI_MarkupPolicy"/>
</GetProperties>
```

o The returned data should include **the policy for Markups**
  - o This is basically an XML representation of rules that dictates various actions related to Markups and based on certain conditions.
  - o For detailed description of markup policy, refer to 'AutoVue Installation and Administration manual":

```
<GetProperties>
      <Property name="CSI_MarkupPolicy">
            <![CDATA[
                  XML content of markup policy
            ]]>
      </Property>
</GetProperties>
```

New in v20 – Public Key:

o Get Properties request may include '**PK**' property.

```
<GetProperties>
   ; Request for public key
   <Property name="PK"></Property>
</GetProperties>
```

- The returned data should include public key used by AutoVue Server to encrypt Authorization block

```
<GetProperties>
      <Property name="PK">
         <![CDATA[public key ]]>
     </Property>
</GetProperties>
```

5 Setting Properties

The AutoVue Server may make requests to set document or collaboration properties. For example:

1. To rename a document.
2. To change a Markup type (between normal, master and consolidated).
3. To notify a collaboration session changed state

The request is described below:

POST:http://www.server.com/servlet/avdms

The Posted data contains XML data of the form:

```
<?xml version="1.0"?>
<!DOCTYPE CSIDATA SYSTEM "csi.dtd">   ; Optional

<CSIDATA>
  <Request>
    ; Original FILENAME (optional)
    <OriginalURL>
      <![CDATA[filename_parameter]]>
    </OriginalURL>
    ; If setting properties for a document
    <CSI_DocID>
      <![CDATA[docID]]>
    </CSI_DocID>
    ; Or if setting properties for a collaboration session
    <CSI_ClbSessionData>
      <![CDATA[dmsClbSessionID]]>
    </CSI_ClbSessionData>
    <Authorization>
      ; Browser cookie
      <Cookie>
        <![CDATA[Cookie]]>
      </Cookie>
      ; DMS session id
      <Property name="CSI_DMSSession">
        <![CDATA[sessionID]]>
      </Property>
      <Property name="name" value="value" />
      <Property name="name" value="value" />
      ; ETC.
    </Authorization>
    <SetProperties>
      ; Setting a single valued property
      <Property name="Property1" value="Value1" />
```

```
      ; Setting a specific index of a multi valued property
      <Property name="Property2" value="Value2" index="index" />
      ; Setting a multi valued property
      <Property name="Property3" />
        <Value value="Value1" />
        <Value value="Value2" />
        <Value value="Value3" />
      </Property>
      ; Notifications from AutoVue
      <CSI_Notifications>
        <Property name="Property1" value="Value1" />
        <Property name="Property2" value="Value2" />
        ; ETC.
      </CSI_Notifications >
      ; ETC.
    </SetProperties>
  </Request>
</CSIDATA>
```

Where ***filename_parameter*** is the **FILENAME** PARAM passed in AutoVue Applet ("JVue") Tag.

Notifications are sent to the DMAPI server component as part of `SetProperties` request.

Predefined property names:

`CSI_Notifications`	Used in `SetProperties` request to notify DMAPI Server component of certain action completed by AutoVue such as `CSI_PagePrinted` and `CSI_DocumentPrinted`.
`CSI_PagePrinted`	Indicates current page number being printed by AutoVue.
`CSI_DocumentPrinted`	Indicates the status of print job. `TRUE` if print job completed successfully or `FALSE` if not.
`CSI_SessionState`	notifies collaboration session state : started or closed
`CSI_UserJoined`	notifies that a user joined a collaboration session
`CSI_UserLeft`	notifies that a user has left the collaboration session
`CSI_DocumentSet`	notifies that the current collaboration session has a new document set. The value is the actual document id as returned by the DMS.
`CSI_MarkupSaved`	notifies a collaboration session that a markup has been saved. The value is the markup id
`CSI_Lock`	Used to lock/unlock documents after check in/out.
`CSI_DMSSession`	Valid session id after successful login to DMS.
`CSI_ClbInitSession`	notifies collaboration session started
`CSI_ClbCloseSession`	notifies collaboration session closed

For example, during printing, AutoVue server would issue `SetProperties` request to the DMAPI server component passing the `CSI_PagePrinted` property with value corresponding to the current page number being printed.

```
      <CSI_Notifications>
```

```
      <Property name="CSI_PagePrinted" value="1" />
   </CSI_Notifications>
```

Once printing is complete, the AutoVue server would issue `SetProperties` request to the DMAPI server component passing the `CSI_DocumentPrinted` property with value indicating the status of the print job: `TRUE` if print job completed successfully or `FALSE` if not:

```
   <CSI_Notifications>
      <Property name="CSI_DocumentPrinted" value="TRUE" />
   </CSI_Notifications>
```

For example, during collaboration, AutoVue server would issue a `SetProperties` request to the DMAPI server component passing the `CSI_ClbInitSession` property with value "*`SessionId`*":

```
   <CSI_Notifications>
      <Property name="CSI_ClbInitSession" value="SessionId" />
   </CSI_Notifications>
```

Then notify a user has joined the session:

```
   <CSI_Notifications>
      <Property name="CSI_UserJoined" value="user1" />
   </CSI_Notifications>
```

That a document has been set:

```
   <CSI_Notifications>
      <Property name="CSI_DocumentSet" value="docID" />
   </CSI_Notifications>
```

Save the markup associated with the currently viewed document in the session:

```
   <CSI_Notifications>
      <Property name="CSI_MarkupSaved" value="DocID" />
   </CSI_Notifications>
```

Then notify a user has left the session:

```
   <CSI_Notifications>
      <Property name="CSI_UserLeft" value="user1" />
   </CSI_Notifications>
```

Once collaboration session is closed, the AutoVue server would issue `SetProperties` request to the DMAPI server component passing the `CSI_ClbCloseSession` property with value "*`SessionId`*":

```
   <CSI_Notifications>
```

```
        <Property name="CSI_ClbCloseSession" value="SessionId" />
    </CSI_Notifications>
```

Note: Collaboration notifications are sent by AutoVue server to DMAPI server component only if Collaboration session was initiated from the DMS (i.e., passing COLLABORATION parameter in AutoVue applet page. For more information, see **'Applet Parameters'** section of the *AutoVue Version Installation and Administration Manual* – install.pdf).

Terminating a DMS Session:

- In order to terminate existing DMS session, the request must include **'CSI_DMSSession'** property with value set to '**0**'. DMS Session id (***sessionID***) must also be included in the XML request:

```
<SetProperties>
 <Property name="CSI_DMSSession" value="0" />
</SetProperties>
```

Locking a file:

- After a file is checked in/out, the file can be locked or unlocked. The request may include '**CSI_Lock**' property. If it is set to **true**, the DM Integration component should place a lock to reserve the document so that no two users can modify same file at the same time. If the file is already locked or reserved by some user, the response should return an error message:

```
<SetProperties>
  ; Lock/Unlock option for file check in/out
  <Property name= "CSI_Lock" value="true|false"/>
</SetProperties>
```

The response is XML structured data:

```
<?xml version="1.0"?>
<!DOCTYPE CSIDATA SYSTEM "csi.dtd">  ; Optional

<CSIDATA>
  <Response>
    <ERROR>
      ; Refer to the section "Error/Authorization Handling"
    </ERROR>
  </Response>
</CSIDATA>
```

In the case of an error (such as insufficient Authorization), an <ERROR> **Response** is returned. Refer to the "**Error/Authorization Handling**" chapter for more information.

6 Saving and Uploading Documents

The AutoVue Server will make requests to upload documents to:

1. Create new Markups.
2. Save modifications to existing markups
3. Upload documents to the DMS database

The request is described below:

POST:http://www.server.com/servlet/avdms

If content type is "**text/xml**", then the POSTed data contains the document specification and authorization information, as well as the file contents, formatted in **XML** as follows:

```
<?xml version="1.0"?>
<!DOCTYPE CSIDATA SYSTEM "csi.dtd">   ; Optional

<CSIDATA>
  <Request>
    ; Original FILENAME (optional)
    <OriginalURL>
      <![CDATA[filename_parameter]]>
    </OriginalURL>
    ; Or if document will be attached to a collaboration session
    <CSI_ClbSessionData>
      <![CDATA[dmsClbSessionID]]>
    </CSI_ClbSessionData>
    <CSI_DocID>             ; Optional for new doc
      <![CDATA[docID]]>
    </CSI_DocID>
    <Authorization>
      ; Browser cookie
      <Cookie>
        <![CDATA[Cookie]]>
      </Cookie>
      ; DMS session
      <Property name="CSI_DMSSession">
       <![CDATA[sessionID]]>
      </Property>
      <Property name="name" value="value" />
      <Property name="name" value="value" />
      ; ETC.
    </Authorization>
    <Properties>
      ; Setting the parent document ( for markup, xrefs )
      <Property name="CSI_BaseDocID">
        <![CDATA[basedocID]]>
      </Property>
```

```
      ; Setting base document name for nested zip file
      <Property name="CSI_BaseDocName" >
        <![CDATA[url_to_another_zip]]>
        <Property name="CSI_BaseDocName" >
                <![CDATA[file_path]]>
        <Property/>
      <Property/>
      ; Setting the document type (markup, rendition … )
      ; use only one at a given time
      <Property name="CSI_RenditionType" value="PCRS_TIF"/>
      <Property name="CSI_MarkupType" value="normal"/>
      <Property name="CSI_ClbDocType" value="chat"/>
      ; Specify if the data is compressed
      <Property name="CSI_Compression" value="false" />
      ; Document type (use only one at a given time)
      <Property name="CSI_DocType"
value="Markup|Rendition|Document|Ressource" subtype ="subtype" />
      ; Lock option for file check-in
      <Property name= "CSI_Lock" value="true|false"/>
      ; Folder id
      <Property name="CSI_Folder">
        <![CDATA[folderID]]>
      </Property>
      ; Setting a single valued property
      <Property name="Property1" value="Value1" />
      ; or Setting a single valued property using CDATA
      <Property name="Property1">
        <![CDATA[Vaule1]]>
      </Property>
      ; ETC.
    </Properties>
    <Save>
      <![CDATA[fileContent_in_BASE64]]>
    </Save>
  </Request>
</CSIDATA>
```

Where ***filename_parameter*** is the **FILENAME** PARAM passed in AutoVue Applet (“JVue”) Tag.

The request may contain a CSI_DocID property. If specified, then the specified document should be overwritten (equivalent to a “Save” command). If not specified, then it is understood that a new document is being created (equivalent to a Save-As command).

The content of the uploaded file is encoded in **BASE64** format. This content must be decoded using BASE64 decoder prior to check-in of the document.

The content of the uploaded file may be compressed in “**gzip**” format, in this case “CSI_Compression” property is set to “true” and file content must be decompressed prior to check-in of the document.

The uploaded data may be a file rendition; in this case, “CSI_RenditionType” indicates the rendition format. For ‘cached meta files’, “CSI_RenditionType” is set to “CSI_META”.
If content type is “**multipart/form-data**”, then the POSTed data contains 2 parts: the “**xml**” part which contains the document specification and authorization information formatted in XML, and the “**file**” part which contains the content of the uploaded file, as follows:

HTTP Request Header:

```
Content Type = multipart/form-data; boundary=CSI_MULTIPART
Content Length = NNN
```

```
--CSI_MULTIPART
Content-Disposition: form-data; name="xml"

<?xml version="1.0"?>
<!DOCTYPE CSIDATA SYSTEM "csi.dtd">   ; Optional
<CSIDATA>
  <Request>
    ; Original FILENAME (optional)
    <OriginalURL>
      <![CDATA[filename_parameter]]>
    </OriginalURL>
    <CSI_DocID>            ; Optional for new doc
      <![CDATA[docID]]>
    </CSI_DocID>
    <Authorization>
      ; Browser cookie
      <Cookie>
        <![CDATA[Cookie]]>
      </Cookie>
      ; DMS session
      <Property name="CSI_DMSSession">
       <![CDATA[sessionID]]>
      </Property>
      <Property name="name" value="value" />
      <Property name="name" value="value" />
      ; ETC.
    </Authorization>
    <Properties>
      ; Setting the parent document ( for markup, xrefs )
      <Property name="CSI_BaseDocID">
        <![CDATA[basedocID]]>
      </Property>
      ; Setting the document type (markup, rendition … )
      ; use only one at a given time
      <Property name="CSI_RenditionType" value="PCRS_TIF"/>
      <Property name="CSI_MarkupType" value="normal"/>
      ; Specify if the data is compressed
      <Property name="CSI_Compression" value="false" />
      ; Setting a single valued property
      <Property name="Property1" value="Value1" />
      ; or Setting a single valued property using CDATA
```

```
      <Property name="Property1">
        <![CDATA[Vaule1]]>
      </Property>
      ; ETC.
    </Properties>
    <Save />
  </Request>
</CSIDATA>
--CSI_MULTIPART
Content-Disposition: form-data; name="file"
Content-Type: text/plain

fileContent_in_BASE64
--CSI_MULTIPART--
```

Where NNN is length in bytes of the 2 parts ('xml' and 'file').

Collaboration:

If your DMS server-side component returns true for **`CSI_ClbSaveChat`** (refer to GetProperties section), then when the host closes the collaboration session, AutoVue server calls `save` with chat content passed in a file.

Request

```
<CSIDATA>
  <Request>
    <CSI_ClbSessionData>
      <![CDATA[dmsSessionIdentifier]]>
    </CSI_ClbSessionData>
    <Authorization>
      <Cookie>
      <![CDATA[cookie]]>
      </Cookie>
    </Authorization>
    <Properties>
      <Property name="CSI_BaseDocID">
        <![CDATA[basedocID]]>
      </Property>
      <Property name="CSI_ClbDocType" value="chat"/>
    </Properties>
    <Save>
      <![CDATA[ChatContent_in_BASE64]]>
    </Save>
  </Request>
</CSIDATA>
```

Note:

- If your DMS server-side component supports multipart/form-data, then the request would be in that format.
- Chat content is base 64 encoded.
- Chat content may be compressed in which case `CSI_Compression` would be set to `true`.

Locking a document:

- Save request may include '**CSI_Lock**' property. If set to **false**, DM Integration component should remove any lock on the document provided the lock was placed by same user.

```
; Lock option for file check-in
<Property name= "CSI_Lock" value="true|false"/>
```

- Save request may also include ***folderID*** ('**CSI_Folder**' property) to specify the location where document is to be checked into DMS:

```
; Folder id
<Property name="CSI_Folder">
  <![CDATA[folderID]]>
</Property>
```

- Save request should include ***basedocID*** ('**CSI_BaseDocID**' property) to maintain parent/child relationship as well as document type such as **CSI_Assembly**, **CSI_Part…**

```
; Setting the parent document (for xrefs, parts… )
<Property name="CSI_BaseDocID">
  <![CDATA[basedocID]]>
</Property>
; Document type (use only one at a given time)
<Property name="CSI_DocType"
value="Markup|Rendition|Document|Ressource" subtype ="subtype" />
```

Valid values are shown below:

value	subtype
Markup	normal master consolidated
Rendition	CSI_META PCRS_TIF
Document	CSI_Assembly CSI_Part CSI_Drawing CSI_Xref
Resource	CSI_Font CSI_Line

The response is XML structured data:

```
<?xml version="1.0"?>
<!DOCTYPE CSIDATA SYSTEM "csi.dtd">  ; Optional

<CSIDATA>
  <Response>
```

```
    <Property name="CSI_DocID">
      <![CDATA[DocID]]>
    </Property>
    <ERROR>
      ; Refer to the section "Error/Authorization Handling"
    </ERROR>
  </Response>
</CSIDATA>
```

The DocID specifies the document ID of the newly created document.

In the case of an error (such as insufficient Authorization), an < ERROR> **Response** is returned. Refer to the "**Error/Authorization Handling**" chapter for more information.

If the Upload fails to the DMS, the AutoVue Server may manage the upload itself.

Distributed Environments:

In a distributed environment where several remote cache servers may be used for storing files, main DMS server component may redirect the **Save** request to another DMS server component that is installed closer to where files actually reside. This greatly improves performance since AutoVue Server is installed at same location as the remote cache server.

Redirection is done by returning a ticket authorizing AutoVue Server to upload the file directly to another location specified in the redirection response.

Prior to Save request, AutoVue Server asks main DMS server component if redirection is supported. If yes, DMS Server component should return '**true**' for GetProperties request of **CSI_Redirected** property.

If **'true**' is returned, Save request made to main DMS server component will not contain any file. In this case, the main DMS server component should return a **ticket** with information of the location where AutoVue should upload the file. XML response is same as Download request.

AutoVue then adds the **ticket** to Authorization block of the Save request made to the DMS server component located at remote cache site.

Once uploaded file is checked in successfully, DMS server component located at remote cache site should return a confirmation in the form of a **receipt** in place of the returned docid.

AutoVue then forwards this receipt to main DMS server component to finalize Save request. Main DMS server component should then return docid of the uploaded file.

When base file is a zipped file:

- In case base file is a zip file, Save request may include '**CSI_BaseDocName**' property. If set, DM Integration component should save uploaded file (markup or metafile) and tag it with base doc name.
- DM Integration component can use this property to identify which markups/metafile belong to which file inside the zip file.
- '**CSI_BaseDocName**' property may be nested in case a zip file contains another zip file.
- In AutoVue, format for a URL to zip file is :

```
url_to_zip//url_to_another_zip//file_path
```

```
<Properties>
   <Property name=”CSI_BaseDocID”>
     <![CDATA[basedocID]]>
   </Property>
   ; Setting base document name for nested zip file
    <Property name="CSI_BaseDocName" >
      <![CDATA[url_to_another_zip]]>
      <Property name="CSI_BaseDocName" >
              <![CDATA[file_path]]>
      <Property/>
    <Property/>
</Properties>
```

- To better support escape, special and foreign characters, Value of a Property inside <Properties> may be sent as CDATA section. DMAPI Server component should always support both cases where value can be either attribute or CDATA.

```
    ; Setting a single valued property using CDATA
    <Property name=”Property1”>
      <![CDATA[Vaule1]]>
    </Property>
```

AutoVue Mobile:

- When synchronizing markups from AutoVue Mobile file, Save request may include '**CSI_OfflineDocAuthor**' property.

```
    ; Offline Author
    <Property name=”CSI_OfflineDocAuthor”>
      <![CDATA[<author_name>]]>
    </Property>
```

- '**CSI_OfflineDocAuthor**' property indicates original author of a markup file stored inside AutoVue Mobile file that is being synchronized with DMS.
  - DM Integration component can use the value of this property to tag uploaded markup file when being imported into DMS.

7 Deleting Documents

The AutoVue Server can make requests to delete documents for example to delete one or more markups attached to a document.

POST:http://www.server.com/servlet/avdms

The POSTed data contains XML data of the form:

```
<?xml version="1.0"?>
<!DOCTYPE CSIDATA SYSTEM "csi.dtd">   ; Optional

<CSIDATA>
  <Request>
    ; Original FILENAME (optional)
    <OriginalURL>
      <![CDATA[filename_parameter]]>
    </OriginalURL>
    <CSI_DocID>                  ; Required
      <![CDATA[docID]]>
    </CSI_DocID>
    <Authorization>
      ; Browser cookie
      <Cookie>
        <![CDATA[Cookie]]>
      </Cookie>
      ; DMS session
      <Property name="CSI_DMSSession">
       <![CDATA[sessionID]]>
      </Property>
      <Property name="name" value="value" />
      <Property name="name" value="value" />
      ; ETC.
    </Authorization>
    <Delete/>
  </Request>
</CSIDATA>
```

Where ***filename_parameter*** is the **FILENAME** PARAM passed in AutoVue Applet ("JVue") Tag.

The specified CSI_DocID corresponds to the ID of a document to be deleted.

The response is XML structured data:

```
<?xml version="1.0"?>
<!DOCTYPE CSIDATA SYSTEM "csi.dtd">  ; Optional
```

```
<CSIDATA>
  <Response>
    <ERROR>
      ; Refer to the section “Error/Authorization Handling”
    </ERROR>
  </Response>
</CSIDATA>
```

In the case of an error (such as insufficient Authorization), an < ERROR> **Response** is returned. Refer to the “**Error/Authorization Handling**” chapter for more information.

8 Error/Authorization Handling

Errors are returned as part of the **<Response>** XML Data.
Unless explicitly specified, the operation is assumed to have succeeded.
Two formats for the <ERROR> response are possible:

- Success

```
<ERROR>
  <SUCCEED  code="code"  message="message">
</ERROR>
```

This indicates that no error occurred. It is optional as success is assumed, if not specified. The code and message attributes are optional.

- Failure

```
<ERROR>
  <FAIL code="code" message="message"/>
  <Authorization>                ; Optional
    <STATIC value="value">
    <EDIT  id="id1" name="name" hidden="false|true" />
    <COMBO id="id2" name="name" default="default">
      <Value value="value1"/>
      <Value value="value1"/>
      <Value value="value1"/>
    </COMBO>
    ; ETC.
  </Authorization>
</ERROR>
```

In case of failure, an optional error code and message can be returned.
Optionally an Authorization body may be returned. This should be created in the case where the user does not have sufficient authorization to perform a certain task.
The body of the <Authorization> element contains the definition of the dialog that will be presented to the user.
Dialog elements supported are:

- STATIC: Attribute “value” contains the text to display.
- EDIT: A single line edit control. The “id” attribute is a unique identifier. The “name” attribute contains the prompt-text to display before the edit control. If the “hidden” attribute is “true” then it behaves like a password entry field (i.e. the text entered is not displayed on the screen).
- COMBO: A single selection list. The “id” field is a unique identifier. The “name” attribute contains the prompt-text to display before the control. The “default” attribute is optional and contains a default selection. The “readonly” attribute controls whether the user can enter values other than those specified.

The “Value” elements that proceed contain default values that will populate the list-box.

When the AutoVue Server gets an Authorization failure error, it forwards the information to the Client. The Client constructs the dialog box to allow the use to enter the necessary authorization data.

The Authorization data entered by the user is then sent in all subsequent "**<Request>**" bodies. The Authorization data is formatted as below:

```
<Authorization>
  <Property name="id1" value="value"/>
  <Property name="id2" value="value"/>
  ; ETC.
</Authorization>
```

The body of the < Authorization> element contains a series of <ID> element children. Each contains two required attributes: "id" which corresponds to the unique "id" specified in the "Response" <ERROR> <Authorization> body and "value" which contains the data entered by the user.

Security:

- EDIT GUI element for passwords has an attribute called ' encrypted',
- If 'encrypted' is set to 'true', then EDIT element should include *encryption* key.
  - In this case, password sent by AutoVue server will be encrypted and DM Integration component should decrypt it before using it.

```
<EDIT  id="ID_PASSWORD" name="Password" hidden="true" encrypted="true|false">
    <PK>
      <![CDATA[encryption_key]]>
    </PK>
</EDIT>
```

9 Encrypting Authorization Block

New in v20:

In order for AutoVue to encrypt authorization block, it sends a request for public key. If a valid public key is returned, AutoVue will use it to encrypt entire Authorization block. The Authorization block in all subsequent requests from AutoVue Server will be encrypted as illustrated:

```
<Authorization   KEY="public key data">
      <![CDATA[encrypted content]]>
</Authorization>
```

In order to activate this feature, following line must be commented out in jvueserver.properties located under <AutoVue install folder>/bin folder

```
dms.vuelink.version=19.3
```

10 Appendix A: DMA API Extension to support Real-Time Collaboration

This appendix describes the Document Management (DM) API's which have been extended to support Real-Time Collaboration for AutoVue.

10.1 GetProperties

- If AutoVue applet is launched with COLLABORATION parameter with VALUE=**"INIT"**, then AutoVue server queries the DMS server-side component for collaboration session id by calling `GetProperties(`**`CSI_ClbSessionID`**`)`. The DMS server-side component can extract the session id from this session data. The session id should be unique.

Request

```
<?xml version="1.0"?>
<CSIDATA>
  <Request>
    <CSI_ClbSessionData>
      <![CDATA[dmsSessionIdentifier]]>
    </CSI_ClbSessionData>
    <Authorization>
      <Cookie>
      <![CDATA[cookie]]>
      </Cookie>
    </Authorization>
    <GetProperties>
      <Property name="CSI_ClbSessionID"></Property>
    </GetProperties>
  </Request>
</CSIDATA>
```

Response

```
?xml version="1.0"?>
<CSIDATA>
 <Response>
  <GetProperties>
      <Property name="CSI_ClbSessionID"  value="SessionId">
      </Property>
  </GetProperties>
  <ERROR>
      <SUCCEED code="code" message="message">
      </SUCCEED>
  </ERROR>
 </Response>
</CSIDATA>
```

- If user selects Collaboration→Invite… or Collaboration→Session Information… from AutoVue Applet menu, AutoVue Server sends the request to DMS server-side component by calling GetProperty(CSI_Collaboration).

Request

```
<?xml version="1.0"?>
<CSIDATA>
  <Request>
    <CSI_ClbSessionData>
      <![CDATA[dmsSessionIdentifier]]>
    </CSI_ClbSessionData>
    <Authorization>
      <Cookie>
      <![CDATA[cookie]]>
      </Cookie>
    </Authorization>
    <GetProperties>
      <Property name="CSI_Collaboration"></Property>
    </GetProperties>
  </Request>
</CSIDATA>
```

Response should include

- a GUI section which includes
  - DisplayOptions section for enabling/disabling GUI items in Invitation dialog box.
  - Display section lists the attributes to be displayed in the Session selection dialog along with the width to reserve for the attributes display. The display width is specified as a number of characters
  - Invitation section lists users who can be invited to the collaboration session.
- a CSI_ClbUsers section lists users who have already been invited to the collaboration.
- a Session section lists session information (properties) such as session title, id, type, subject, duration, start time…etc.
- CSI_ClbSaveChat indicates whether DMS server-side component supports saving chat transcript.

Response

```
<?xml version="1.0"?>
<CSIDATA>
 <Response>
   <GetProperties>
    <Property name=”CSI_Collaboration”>
        <Property name = “GUI”>
          <Property name = “DisplayOptions”>
            <Property name=“AllowAdd” value=”true|false” />
            <Property name=“AllowAddNew” value=”true|false”/>
            <Property name=“AllowRemove” value=”true|false”/>
            <Property name=“AllowLayerColor” value=”true|false”/>
          </Property>
          <Property name = “Display”>
            <Property name=”name1” value=”width” />
            <Property name=”name2” value=”width” />
          </Property>
          <Property name = “Invitation”>
            <List  id=”CSI_ClbUsers” name=”user””>
```

```
              <Value value=”value1”/>
              <Value value=”value1”/>
              <Value value=”value1”/>
            </List>
          </Property>
        </Property>
        <Property name = “CSI_ClbUsers”>
           <Value value =”user1”/>
           <Value value =”user2”/>
        </Property>
        <Property name = “Session”>
          <Property name="CSI_ClbSessionID">
               <![CDATA[collaborationSessionID]]>
          </Property>
          <Property name= ”CSI_ClbSessionType” value=”private|public”/>
          <Property name= ”CSI_ClbSaveChat” value=”true|false”/>
          <Property name= ”name1” value=”value1”/>
          <Property name= ”name2” value=”value2”/>
        </Property>
      </Property>
    </GetProperties>
  </Request>
</CSIDATA>
```

10.2 SetProperties

Various notifications related to Real-Time Collaboration have been added to DMAPI. AutoVue Server sends these notifications by calling SetProperties(NotificationId) request.

- When a collaboration session starts, AutoVue server calls SetProperties(CSI_ClbInitSession).

Request

```
<?xml version="1.0"?>
<CSIDATA>
  <Request>
    <CSI_ClbSessionData>
      <![CDATA[dmsSessionIdentifier]]>
    </CSI_ClbSessionData>
    <Authorization>
      <Cookie>
      <![CDATA[cookie]]>
      </Cookie>
    </Authorization>
    <SetProperties>
      <CSI_Notifications>
      <Property name="CSI_ClbInitSession" value="SessionId"></Property>
      </CSI_Notifications>
    </SetProperties>
  </Request>
</CSIDATA>
```

- When a user joins the collaboration session, AutoVue server calls SetProperties(CSI_UserJoined).

Request

```
<?xml version="1.0"?>
<CSIDATA>
  <Request>
    <CSI_ClbSessionData>
      <![CDATA[dmsSessionIdentifier]]>
    </CSI_ClbSessionData>
    <Authorization>
      <Cookie>
      <![CDATA[cookie]]>
      </Cookie>
    </Authorization>
    <SetProperties>
      <CSI_Notifications>
      <Property name="CSI_UserJoined" value="UserName"></Property>
      </CSI_Notifications>
    </SetProperties>
  </Request>
</CSIDATA>
```

- When a user leaves the collaboration session, AutoVue server calls SetProperties(CSI_UserLeft).

Request

```
<?xml version="1.0"?>
<CSIDATA>
  <Request>
    <CSI_ClbSessionData>
      <![CDATA[dmsSessionIdentifier]]>
    </CSI_ClbSessionData>
    <Authorization>
      <Cookie>
      <![CDATA[cookie]]>
      </Cookie>
    </Authorization>
    <SetProperties>
      <CSI_Notifications>
      <Property name="CSI_UserLeft" value="UserName"></Property>
      </CSI_Notifications>
    </SetProperties>
  </Request>
</CSIDATA>
```

- When host switches to another document to collaborate on, AutoVue server calls SetProperties(CSI_DocumentSet).

Request

```
<?xml version="1.0"?>
<CSIDATA>
  <Request>
    <CSI_ClbSessionData>
      <![CDATA[dmsSessionIdentifier]]>
    </CSI_ClbSessionData>
    <Authorization>
      <Cookie>
      <![CDATA[cookie]]>
      </Cookie>
    </Authorization>
    <SetProperties>
      <CSI_Notifications>
      <Property name=" CSI_DocumentSet" value="DocId"></Property>
      </CSI_Notifications>
    </SetProperties>
  </Request>
</CSIDATA>
```

- When host saves the markup for the collaboration session, AutoVue server calls SetProperties(CSI_MarkupSaved).

Request

```
<?xml version="1.0"?>
<CSIDATA>
  <Request>
    <CSI_ClbSessionData>
      <![CDATA[dmsSessionIdentifier]]>
    </CSI_ClbSessionData>
    <Authorization>
      <Cookie>
      <![CDATA[cookie]]>
      </Cookie>
    </Authorization>
    <SetProperties>
      <CSI_Notifications>
      <Property name=" CSI_MarkupSaved " value="MarkupDocName"></Property>
      </CSI_Notifications>
    </SetProperties>
  </Request>
</CSIDATA>
```

- When a collaboration session ends, AutoVue server calls SetProperties(CSI_ClbCloseSession).

Request

```
<?xml version="1.0"?>
<CSIDATA>
  <Request>
    <CSI_ClbSessionData>
      <![CDATA[dmsSessionIdentifier]]>
    </CSI_ClbSessionData>
    <Authorization>
      <Cookie>
      <![CDATA[cookie]]>
      </Cookie>
    </Authorization>
    <SetProperties>
      <CSI_Notifications>
      <Property name="CSI_ClbCloseSession" value="SessionId"></Property>
      </CSI_Notifications>
    </SetProperties>
  </Request>
</CSIDATA>
```

10.3 Save

If your DMS server-side component returns true for CSI_ClbSaveChat (refer to GetProperties section), then, when host closes the collaboration session, AutoVue server calls save with chat content passed in a file.

Request

```
<CSIDATA>
  <Request>
    <CSI_ClbSessionData>
      <![CDATA[dmsSessionIdentifier]]>
    </CSI_ClbSessionData>
    <Authorization>
      <Cookie>
      <![CDATA[cookie]]>
      </Cookie>
    </Authorization>
      <Property name="CSI_BaseDocID">
        <![CDATA[basedocID]]>
      </Property>
      <Property name="CSI_ClbDocType" value="chat"/>
    </Properties>
    <Save>
      <![CDATA[ChatContent_in_BASE64]]>
    </Save>
  </Request>
</CSIDATA>
```

Note:

- If your DMS server-side component supports multipart/form-data, then the request would be in that format.
- Chat content is base 64 encoded.
- Chat content may be compressed in which case CSI_Compression would be set to true.

10.4 AutoVue Applet Page

A parameter called ‘COLLABORATION’ is available on the AutoVue applet.

- To Initiate a session

```
<param NAME=COLLABORATION
            VALUE="INIT:CSI_ClbDMS=dmsIndex;
            CSI_ClbSessionData=123456789;
            CSI_ClbSessionSubject=Subject;
            CSI_ClbSessionType=public|private;
            CSI_ClbUsers=user1,user2">
```

- To Join a session:

```
<param NAME=COLLABORATION
            VALUE="JOIN:CSI_ClbDMS=dmsIndex;
            CSI_ClbSessionData=123456789">
```

Where,

`CSI_ClbDMS`	Represents the URL to DMS server-side component index. Ex.: `http://myserver/servlet/DMS`
`CSI_ClbSessionData`	Represents the handle for the DMS collaboration session. This may include session id plus context information which may later be needed by DMS server-side component. **Note:** AutoVue server passes this to the DMS server-side component for collaboration related requests.
`CSI_ClbSessionSubject`	Represents the session subject
`CSI_ClbSessionType`	Either public or private
`CSI_ClbUsers`	Comma separated list of invitees to the meeting

Note:

- CSI_ClbSessionData should not include semicolon “;”. AutoVue uses semicolon as a separator.
- When joining a session, FILENAME param should be omitted.
- Whenever AutoVue server queries for DMS collaboration, it passes the CSI_ClbSessionData.

11 Appendix B: DMS Arguments in AutoVue Applet Page

A parameter called 'DMSARGS' lets you pass DMS arguments as applet parameters (Key/value pair) when launching the AutoVue applet. The key/value pairs are then passed back to DMS component by the AutoVue server as part of posted data in each DMAPI request.

- DMS Arguments as applet parameters

```
<PARAM NAME="DMSARGS"        VALUE="DMS_key1;DMS_key2;DMS_key3;...DMS_keyN">
<PARAM NAME="DMS_key1"       VALUE="value1">
<PARAM NAME="DMS_key2"       VALUE="value2">
<PARAM NAME="DMS_key3"       VALUE="value3">
...
<PARAM NAME="DMS_keyN"       VALUE="valueN">
```

Where,

DMSARGS	A semi-colon ';' separated list of DMS argument names. Ex.: DMS_HOST;DMS_LIBRARY;DMS_VAULT;DMS_TICKET
DMS_key1=value1 DMS_key2=value2 … DMS_keyN=valueN	Key/value pairs of DMS arguments.

- DMS Arguments passed back to DMS component as properties

```
<Properties>
<Property name="DMS_key1"><![CDATA[value1]]></Property>
<Property name="DMS_key2"><![CDATA[value2]]></Property>
<Property name="DMS_key3"><![CDATA[value3]]></Property>
...
<Property name="DMS_keyN"><![CDATA[valueN]]></Property>
</Properties>
```

Note:

- Values are packaged in CDATA section to allow for escape characters.

Predefined DMSARGS names:

DMS_PRESERVE_COOKIES	Set this parameter to "true" if your application server where DMAPI server component is deployed is setup in a cluster environment. Otherwise, set to "false" (Default is "false"). Example: <PARAM NAME="DMSARGS" VALUE="DMS_PRESERVE_COOKIES"> <PARAM NAME="DMS_PRESERVE_COOKIES" VALUE="TRUE">

12 Feedback

Oracle products are designed according to your needs. We appreciate your feedback, comments or suggestions. Contact us by e-mail or telephone.

12.1 General Inquiries

Telephone: +1.514.905.8400
E-mail: autovuesales_ww@oracle.com
Web Site: http://www.oracle.com/autovue/index.html

12.2 Sales Inquiries

Telephone: +1.514.905.8400
E-mail: autovuesales_ww@oracle.com

12.3 Customer Support

Web Site: http://www.oracle.com/autovue/index.html