
# API GUIDE

Revision 5.2.13

# Copyright Information

This description is for information purpose only.

---

BPO Management Services (BPOMS), reserves the right to alter this description or to adapt it to technical conditions at any time. BPOMS shall not be liable for any damage caused by the contents of this document or the use of the described product.

BPOMS provides this publication "as is'" without warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability or fitness for a particular purpose. BPOMS has the sole copyright for this document. Any reproduction, adaptation, or compilation work for own use and / or distribution purposes is allowed with the written consent of BPOMS only.

All brand and product names in this publication are registered trademarks and trademarks of their respective holders.


BPO Management Services
1290 N Hancock Street
Anaheim Hills, CA 92807 USA
Phone: (714) 974-2670
Toll Free 1-888-376-8900
Fax: (714) 970-1342

eMail: info@ereviewonline.com

# Table of Contents

# Introduction

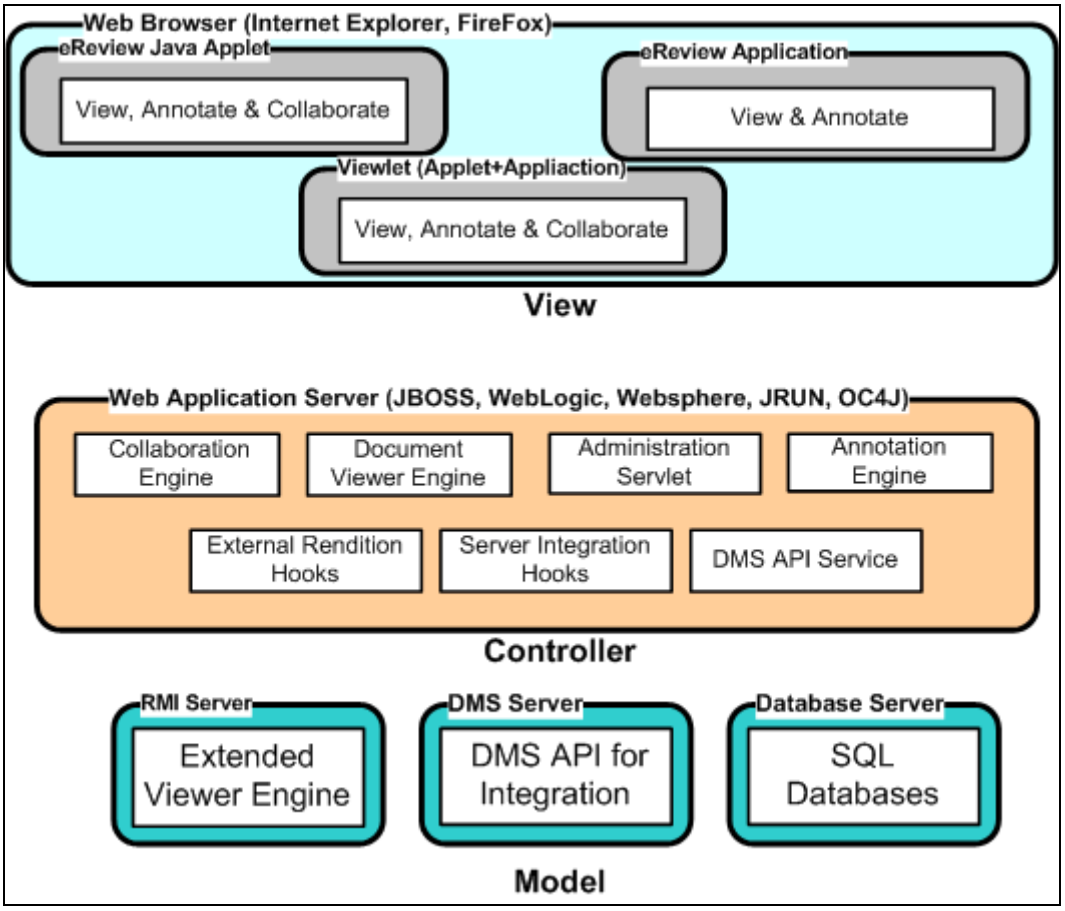
The eReview API set enables programmers to integrate a Java based viewer. The Java based viewer is accessible from any web browser on any platform. eReview allows you to view and markup many different types of documents and drawings without having or using the application that created them. Markups and annotations are non-destructive; i.e., the original document is never changed; rather, the markups, annotations and comments are saved in a corollary file, providing a history of suggested changes. This process prevents uncontrolled changes to a document/drawing while protecting the integrity of the document/drawing under review.

Any project or product lifecycle includes multiple reviews and comments on in-process files. To facilitate reviewing files, one needs to view and mark-up many different document types online, such as spreadsheets, word-processor files, PDF files, CAD drawings and scanned images. Organizations have long felt the need of a viewing and markup tool to work with in-process drawings and documents without the original software. There are viewers available today those allow you to work with many file formats, including Microsoft Office documents, CAD drawings (such as DWG, DXF, DWF, DGN), Portable Document Formats (PDF), Scanned Raster Images ( such as TIFF, PNG, GIF, and JPEG), all from within a single thin client application in web browser interface. However, these products still at best offer what can be termed as sequential or asynchronous document review.

The new paradigm of real-time collaboration eliminates the ineffectiveness of knowledge sharing due to the nature of information media and its physical location and allows users to effortlessly unlock and capture content residing in documents, drawings, CAD models, paper, multimedia presentation and animations, and even tacit knowledge residing in the human mind (ideas and feedback). BPOMS offers organizations a synchronous as well as an asynchronous document viewing, annotation, printing, collaboration and workflow where its users, employees, customers, vendors, suppliers and partners anywhere across the globe are able to view data and documents in real-time, make markups to documents, co-view and co-markup with others, chat, share desktops and applications, carry out audio conference all from within one integrated environment. The obvious benefits of such collaboration platforms are improved access to information, effective document-centric meetings, reduced travel costs, enhanced teamwork, and ability to store the ad-hoc ideas and innovations for future use.

# Chapter 1: System Overview

eReview is based on Model-View-Controller ("MVC") architectural design pattern. It organizes an interactive application into three separate modules: one for the application **Model** with its data representation and business logic, the second for **Views** that provide data presentation and user input, and the third for a **Controller** to dispatch requests and control flow. The following figure describes eReview MVC architecture in detail**.**




Server Integration Hooks are mostly utilized to extend the integration capabilities by incorporation DMS API. The DMS APIs for Integration are used to authenticate, checkout and check-in documents from the vault/repository. In many implementations in absence of DMS API layer a customized code could be written to implement integration directly with SQL Database. JDBC driver connection are established in the extended class to save markup and meeting related information for a selected document viewed using eReview.

eReview supports both server and client side viewing paradigm. In case of non windows based applications servers some of the viewer library developed as Windows component could be places in a windows server and access the service through RMI base connection. Such non windows servers where eReview deployed are UNIX, LINUX, HUUX, AIX or AS400. As eReview server side component is a JAVA based, it makes the deployment easy on any platform.

# Chapter 2: System Requirements

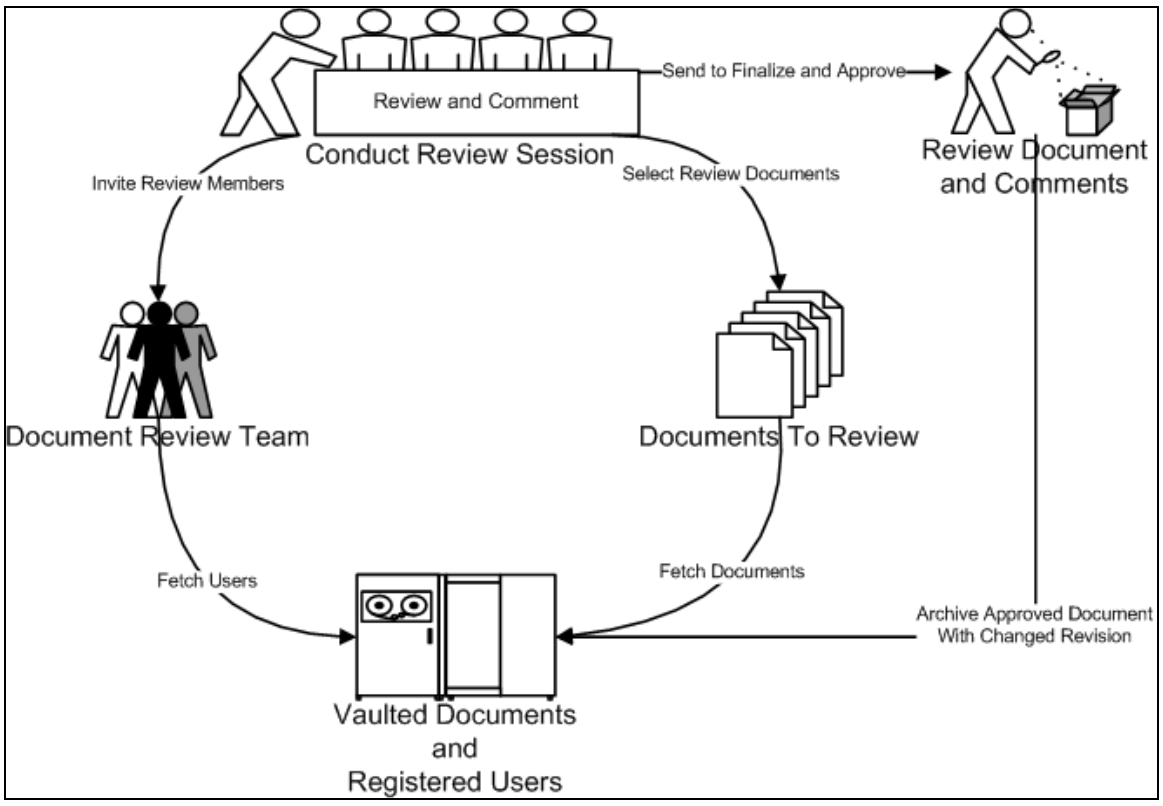
The eReview application is developed using Java 2 technology. On the server side it requires a Web Application Server. The standard deliverable of eReview application is a ereview.war file. This is deployed on a WAS to install few Servlets to process HTTP requests. eReview applet on the other hand is loaded into a web browser. Applet and Servlet together establish a collaborative framework. To execute many view and annotation operations applet instance on the browser sends HTTP requests and receives data through as HTTP response objects.

A minimum system requirement to host and use eReview viewer is listed as follows.

| Environment | eReview Requirement |
|---|---|
| **Web Application Server** | **Tomcat / Jboss / WebLogic / WebSphere** |
| **Operating System** | Windows NT 4.0 Server or later, Windows 2000-Advanced Server, XP, HP-UX version 11 or later, Sun Solaris 2.6 or later, Red-hat LINUX 2.1 or later. |
| **Processor** | Pentium IV at 1GigZ or higher (or equivalent processor) recommended |
| **Hard Disk Space** | 500MB |
| **RAM** | 1GB or more recommended |
| **JRE and additional components** | JRE 1.4.2 or later. |
| **e-mail Service(optional)** | To send and receive eReview meeting invitations SMTP email server IP address is required. |
| **Client** | **Any Web Browser with Java Runtime Engine 1.4.2** |
| **Operating System** | Windows 2000, 2003, Windows XP, Sun Solaris 2.6 or later, HP-UX version11, Sun Solaris 2.6 or later, Red-hat LINUX 2.1 or later. |
| **Hard Disk Space** | 5.5 MB. For optional client side viewing for faster performance: 60 MB. |
| **RAM** | 256 MB or more recommended |
| **Software Plug-ins** | JRE 1.4.2 or later. |

# Chapter 3: Integration Requirements

Document is the primary asset of any Enterprise Portal. Many ECM/PDM/PLM systems in the industry handle document reviewing and commenting and revise the documents every day. Most important task is to view and apply the footprint that you have created through certain review process using viewer available annotation capabilities. Seamless viewer integration with a Document Review process involves authenticated user access to the repository. Create annotations and apply on the document through well formed relationships available through the document management library. The annotations and comments are saved into a corollary file of the base document inside the repository. A reviewer at any point can load or unload different document revisions into the viewer to check his previously created comments or annotations. A genuine document review process could be seen as described in following image.




As far as the integration development is concerned you would need a Web Application Server to host eReview Servlets. Deploy ereview.war file and make sure you have a valid demo license applied to the deployment. Load ereview application home page on your web browser and check if the JRE plug-ins are installed. The homepage has links to JRE installable distribution for the end user to install and make the browser ready to load eReview viewer applet.

# Chapter 4: Viewer Integration

Most of web based application read the documents from the server hard drive. Integration with eReview Viewer simplifies the document viewing operation with a support for multiple document types in single applet instance. The integration requires embedding the eReview applet in a HTML page and specifying the document path on the server as one of applet tag parameter. eReview applet send the request to eReview Integration Servlet to stream down the document content to display on one of eReview MDI child frame window. It is mandatory to associate User Identification to eReview viewer session. This information is also provided to the eReview applet through applet tag parameters. The annotation created in each viewing session is marked with the user Identification. This action simplifies challenges in conducting collaborative view and annotation operation. The markups or annotations are traced by the attendee ID and color selected. The HTML page generation with applet tag could be done in one of following approaches.

## 4.1 Applet Tags In HTML page

Use following sample <applet/> tag with appropriate applet tag values to embed eReview applet in html pages and view the document specified in **docPath** parameter.

```
<applet height="100%" width="100%" name="eReviewApplet"
code="chat.applet.CHttpChatApplet.class"  codebase="http://localhost:8080/ereview"
width=95% height=95%  >
<param name='type' value='application/x-java-applet;version=1.5'>
<param name='image' value='images/logo.gif'>
<param name='codebase' value='.'>
<param name='code' value='chat.applet.CHttpChatApplet.class'>
<param name='server_url' value='EReviewServlet'>
<param name='IntegrateDMSCGI' value='CIntegrateServlet'>
<param name='sock_emul_url' value='CSockEmulServlet'>
<param name='boxmessage' value='Loading eReview Applet...'>
<param name='boxbgcolor' value='255,255,255'>
<param name='boxfgcolor' value='192,192,192'>
<param name='progresscolor' value='10,0,255'>
<param name='progressbar' value='true'>
<param name='archive' value='EReviewLite.jar'>
<param name='scriptable' value='true'>
<param name='console' value='+ALL'>
<param name='MeetingTitle' value='ViewOnly'>
<param name='SERVER_TYPE' value='NOTUNIX'>
<param name='JOIN' value='true'>
<param name='MODE' value='VIEW_ONLY'>
<param name='EREVIEW' value='VC'>
<param name='MEETING_ID' value='15-UE'>
<param name='USE_LOCAL_DLL' value='FALSE'>
<param name='load_GUI_conf_path' value='profile_user_Beginner View Only.xml'>
<param name='MKPEDIT' value='false'>
<param name='ShowOpenButton' value='false'>
<param name='BRIEFCASE' value='true'>
<param name='USERFOLDER' value=' C:\\DocumentFolder'>
<param name='docPath' value='C:\\DocumentFolder\\document.tiff'>
<param name='USERNAME' value='Ashok Jena'>
```

<param name='**USERID**' value='127_0_0_1-Ashok_Jena'>
<param name='**AUTODETECT**' value='true'>
</applet>

Few optional parameters are as listed in following sections could be used to execute some advanced operations.
<param name='CURRENT_PAGE ' value='1'>
<param name='OVERLAYTEXT ' value='My Overlay Test Message Here'>
<param name='REDACTION ' value='x,y,w,h'>

Change above bold marked applet tags as directed below to reconfigure eReview applet functionality.

#### 4.1.1 Auto detect file type

Set "AUTODETECT" value to true or false to enable the document auto detection to view using right viewer filter.
Example - A pdf file could be having extension ppt by mistake. Using eReview auto detection mechanism the file is viewed in PDF filter. Otherwise it would be viewed as a Microsoft Power Point file and applet would fail to open the file.

#### 4.1.2 Change user Identification

Set "USERNAME" and "USERID" value with new user name and id to initiate eReview viewer session with different identification. All annotations created will be owned by the respective user.

#### 4.1.3 View specific document from server

Set "docPath" value with the document path on the server. This will allow you to load different documents from the server in your web browser.

#### 4.1.4 Enable Open tool button

To open multiple documents from a specific folder on the server you need to enable "Open" button in your viewer. To enable Open button you need to set following applet tags.
Set "USERFOLDER" with the absolute path to the folder on the server.
Set "BRIEFCASE" to true. This enables to read folders and subfolders in it.
Set "ShowOpenButton" to true. This enables the "Open" tool button.

#### 4.1.5 Enable Markup tools

To enable markup tool for the end user you can specify value to "load_GUI_conf_path". All configurable profiles are found in ereview\WEB-INF\profiles folder. This is a simple XML file and editable to add or remove tool buttons.

#### 4.1.6 Use accelerator DLLs on Client Machine

User accelerator DLLs are mostly required if the eReview is deployed on a UNIX or LINUX based server. Some of the viewer filters depend on windows dlls, so at the client end we download this accelerator dlls and view the document from client hard disk.

Set “USE_LOCAL_DLLS” value to enable accelerator dlls.

#### 4.1.7 Change code base of EReviewNew.jar

Set codebase=http://localhost:8080/ereview to some different http location to load EReviewNew.jar. Most of the time localhost is replaced by the public facing IP address to load eReview applet from any external server.

## 4.2 Request with HTTP Header Parameters

As an alternative to 4.1 there are few HTTP request and response implementation in eReview server to generate HTML pages based on the request headed parameters. HTTP requested header keys are described in the following table:

Send request to http://localhost:8080/ereview/CIntegrateServlet

| Header Key | Default Value | Description |
|---|---|---|
| Attendee_ID | User identification Code | This identifies user applet session |
| Attendee_Name | User Name to Identify Markups | Markup layer created is associated with this name. |
| Meeting_ID | 15-UE | Standard Viewer Session Identifier. Keep the value 15-UE all the time. |
| MEETINGTITLE | ViewOnly Title | Set a name that has to be displayed on web browser title to identify the viewer session |
| Document_Path | Document Path to view | This is the document path on the server. |
| DocumentTitle | Title of the Document | To Display a different title of the document other than the filename itself |
| MKPEDIT | False | This could be set to true or false to enable markup tools or disable markup tools. |
| READONLY_ACCESS | True | A preloaded markup will have read only access for the end user. It restricts the users from edit current markups. |
| Mkpfile_load | False | Set this to true for enabling auto load markup file. |
| Markups_Path | Markup file Path | Specify the Markup Flie path to be viewed along with the document |
| Profile | Profile file name to load | Name of the profile file from web-inf\profiles folder |
| OverlayText | Text to overlay on the document | A text messages could be provided to display on the document as watermark |
| Page | Page number | To load a specific page number from a multiple page document. |
| Redaction | x,y,w,h | Specify start point X and Y coordinate. The specify the width and height |

The base URL for the HTTP request is as given below.
http://localhost:8080/ereview/CIntegrateServlet?REQUEST-TYPE=OPEN_VIEW_ONLY_APP

The header key REQUEST-TYPE=OPEN_VIEW_ONLY_APP identifies to start eReview applet in Viewer Mode. Mandatory additional header keys are specified as follows.
&Attendee_ID=MyName
&Meeting_ID=15-UE
&MEETINGTITLE=ViewOnly
&Attendee_Name=MyName

Following additional header keys are provided to start different instances as specified.

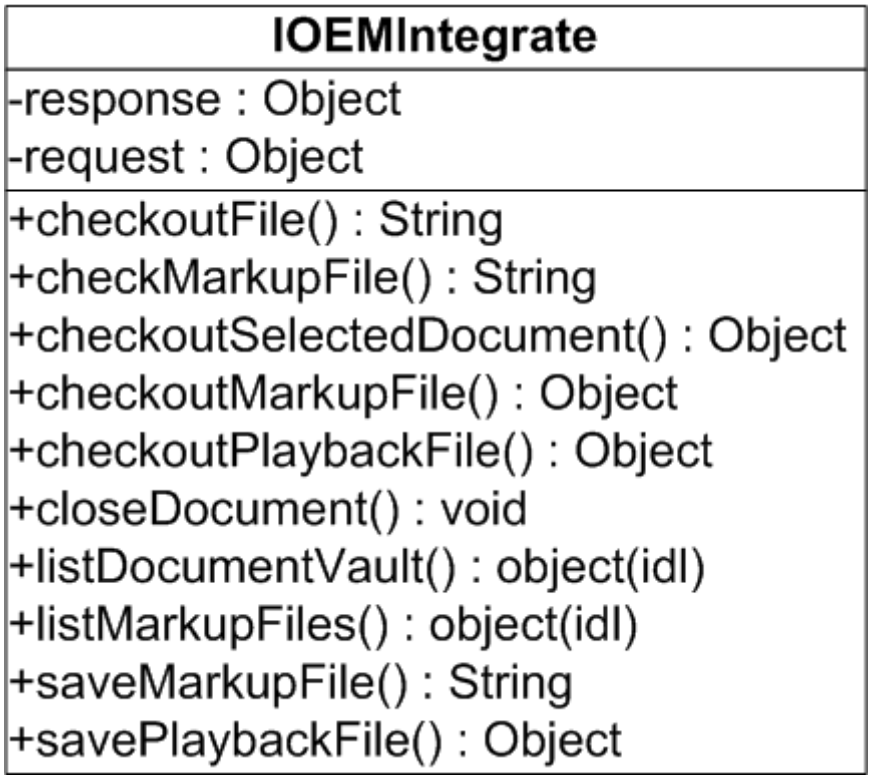
4.2.1 Start the Viewer without markup tools
&MKPEDIT=false
4.2.2 Start the Viewer with markup tools
4.2.3 Start the Viewer with preloaded document and markup files.
4.2.4 Start the Viewer with predefined user profile

# Chapter 5: Viewer Integration Extension

In Chapter 4 it was assumed that the documents are available on server side hard disk. Most of Enterprise Content Management (ECM) application keeps documents encrypted and vaulted in a ORACLE or MS SQL or DB2 database servers. In these cases it is necessary to authenticate using published APIs and checkout to cache the document on server side hard disk before streaming the document to the client side applet. eReview server side integration hooks are designed to extend and implement the checkout and check-in process to read and write documents into ECM vault. Following section explains the steps in detail to implement Viewer Integration Extension.

## 5.1 Implement Extension Class

Extension class is implemented to manage the integration related operation for eReview viewer specific events. There are **nine** such viewer specific events triggered from the browser interface hosting eReview applet to view and annotate on enterprise documents.
In this section implementation of Viewer Specific Events are explained in detail for the developers to extend and implement. A basic java language programming experience is required for the developers.
eReview framework provides an abstract class “IOEMIntegrate” to extend and implement following public interfaces. Each interface is dedicated to one of such nine viewer specific events. The following figure depicts the class diagram of IOEMIntegrate object. This class is placed in EReviewServlet.jar found under web-inf\lib folder of ereview.war deployment.




Look at the sample implementation in SampleImplementation.java delivered along with the ereview.war deployment. Following table explains the each interface name, arguments, return values and its minimal implementation in the sample code. The following table explains how to implement each of the interfaces in the IOEMIntegrate class.

| Interface Name and Arguments | Implementation Detail |
|---|---|
| **checkoutFile(Properties intgInfo)**<br><br>**Interface Argument:**<br>intgInfo- A Java Properties object that holds all the parameters required for the document integration.<br><br>**Return Value:**<br>Absolute path to the document is returned for the applet to view the document.<br><br>Few other parameters like “Profile” for the user UI and markup file path to preload on the document are set to the intgInfo to return back to the applet. | public String checkoutFile(Properties intgInfo) throws Exception<br>{<br>String szCatchedFilePath = "C:\\MyDocPath\\MyDoc1.txt";<br>//To assign custom profile for the end user applet.<br>String userProfile=request.getParameter("Profile");<br>if (userProfile != null) {<br>this.setProperty("Profile",userProfile);<br>}<br>//Set markup files to preload<br>String szMkpPath = "C:\\MyDocPath\\Markups\\MyDoc1_Markup.mkp";<br>intgInfo.setProperty("Markups_Path", szMkpPath);<br>return szCatchedFilePath;<br>} |

| | |
|---|---|
| **checkMarkupFile(Properties intgInfo)**<br><br>**Interface Argument:**<br>intgInfo- A Java Properties object that holds all the parameters required for the check markup file existence integration.<br><br>**Return Value:**<br>A Java Properties object is returned with two parameters.<br>If "File-Extests" is set to true, then eReview applet would display a warning message to overwrite the current markup file.<br>The "server_temp_dir" parameter is used to locate the path where the markup file would be saved. | public Properties checkMarkupFile(Properties intgInfo)<br>{<br>Properties retProp=new Properties();<br>String szMkpFilePath = intgInfo.getProperty("MKPDOCPATH");<br>String szMkpFileSavePath = szMkpFilePath;<br>File fMkp=new File(szMkpFilePath);<br>if (!fMkp.exists()){<br>//The temp path where markup file needs to be saved<br>szMkpFileSavePath = "C:\\MyDocPath\\Markups";<br>}<br>String fileExists=""+fMkp.exists();<br>retProp.setProperty("File-Exists",fileExists);<br>retProp.setProperty("server_temp_dir",szMkpFileSavePath);<br>return retProp;<br>} |
| **checkoutSelectedDocument(Properties intgInfo)**<br><br>**Interface Argument:**<br>intgInfo- A Java Properties object that holds all the parameters required to checkout a selected document from the applet side document open dialog box.<br><br>**Return Value:**<br>A Java Properties object is returned with two parameters.<br>The "filepath" is set to the absolute path of the document selected. If a markup file is available for preload, then "Markups_Path" is set to the absolute path of the markup. | public Properties checkoutSelectedDocument(Properties intgInfo)<br>{<br>Properties retProp=new Properties();<br>String szDocumentId = intgInfo.getProperty("docid");<br>String szDocumentPath = "C:\\MyDocPath\\MyDoc1.txt";<br>String szMarkupPath = "C:\\MyDocPath\\Markups\\MyDoc1_Markup.mkp";<br>//Set Markups_Path with markup file paths if this needs to be loaded along with the document<br>this.setProperty("Markups_Path", szMarkupPath);<br><br>retProp.setProperty("filepath",szDocumentPath);<br>return retProp;<br>} |
| **checkoutMarkupFile(Properties intgInfo)**<br><br>**Interface Argument:**<br>intgInfo- A Java Properties object that holds all the parameters required for checking out the markup file from the vault through integration process.<br><br>**Return Value:**<br>A Java Properties object is returned | public Properties checkoutMarkupFile(Properties intgInfo)<br>{<br>Properties retProp=new Properties();<br>String szDocumentId = intgInfo.getProperty("docid");<br>String szMarkupPath = "C:\\MyDocPath\\Markups\\MyDoc1_Markup.mkp";<br><br>retProp.setProperty("markupfilepath",szMarkupPath); |

| | |
|---|---|
| with two parameters.<br>The “markupfilepath” is set to the absolute path of the markup file selected for the current document in focus. A markup identifier is set to the “docid” to uniquely identify the markup file. | return retProp;<br>} |
| **checkoutPlaybackFile(Properties intgInfo)**<br><br>**Interface Argument:**<br>intgInfo- A Java Properties object that holds all the parameters required for checking out playback file path for a saved meeting session in the integration.<br><br>**Return Value:**<br>A Java Properties object is returned with two parameters.<br>The “playbackfilepath” is set to the absolute path of the playback XML file selected meeting. | public Properties checkoutPlaybackFile(Properties intgInfo)<br>{<br>Properties retProp=new Properties();<br>String szPlaybackPath = "C:\\MyMeetingPath\\Meeting1\\Meeting1.xml";<br><br>retProp.setProperty("playbackfilepath",szPlaybackPath);<br>return retProp;<br>} |
| **closeDocument(Properties intgInfo)**<br><br>**Interface Argument:**<br>intgInfo- A Java Properties object that holds all the parameters required for the close document integration.<br><br>**Return Value:**<br>This interface does not return any value, simply used to clean the cached document for the integration. | public void closeDocument(Properties intgInfo)<br>{<br>//You can delete the document from the cache location<br>} |
| **listDocumentVault(Properties intgInfo)**<br><br>**Interface Argument:**<br>intgInfo- A Java Properties object that holds all the parameters required for getting the list of the documents when user expands the document tree structure in document open selection dialog box for integration.<br><br>**Return Value:**<br>An array of Java Properties object is returned to populate the tree structure in eReview Document Open dialog box. Each of the properties object | public Properties[] listDocumentVault(Properties intgInfo)<br>{<br>Properties[] retValues=null;<br>ArrayList documentFilesList = new ArrayList();<br>//Collect document files list from the vault<br>String szDocumentId = intgInfo.getProperty("docid");<br>String szUserId = intgInfo.getProperty("Attendee_ID");<br>String sz_DocumentPath=intgInfo.getProperty("DOC_PATH");<br>//Lets say there are markup files saved into a specific folder.<br>String szReadDocumentFrom = sz_DocumentPath+"\\"+szUserId; |

| | |
|---|---|
| holds following key and value pairs.<br>title = document name to be displayed in tree leaf nodes.<br>path = the absolute document path will be loaded when selected.<br>size = to display the size of the file next to the leaf name.<br>docid = to identify the document uniquely in the integration.<br>isFolder = to identify if the node could be a leaf node or and node that could be expanded further. | File dir=new File(szReadDocumentFrom);<br><br>File[] listFiles= dir.listFiles();<br>if(listFiles != null)<br>{<br>File curent_file= null;<br>String szDocTitle = "";<br>Properties currentDocProperties = null;<br>String szDocPath = "";<br>String szDocId = "";<br>String szDocFileSize = "";<br>for(int i= 0; i< listFiles.length; i++){<br>currentDocProperties = new Properties();<br>curent_file= listFiles[i];<br>szDocTitle = curent_file.getName();<br>//If you set this to "dummy", then the file will be read when selected from the list at the applet end.<br>szDocPath = curent_file.getAbsolutePath();<br>szDocId = String.valueOf(i);<br>szDocFileSize = String.valueOf(curent_file.length());<br>currentDocProperties.setProperty("title", szDocTitle);<br>currentDocProperties.setProperty("path", szDocPath );<br>currentDocProperties.setProperty("size", szDocFileSize);<br>currentDocProperties.setProperty("docid", szDocId);<br>currentDocProperties.setProperty("isFolder", "false");<br><br>documentFilesList.add(currentDocProperties);<br><br>}<br>}<br>int totalNodeCount = documentFilesList.size();<br>retValues = new Properties[totalNodeCount];<br>for ( int nodeCount = 0; nodeCount < totalNodeCount; nodeCount++){<br>retValues[nodeCount] = (Properties)documentFilesList.get(nodeCount);<br>}<br><br>return retValues;<br>} |
| **listMarkupFiles(Properties intgInfo)**<br><br>**Interface Argument:**<br>intgInfo- A Java Properties object that holds all the parameters required for getting the list of the markups when user expands the markup tree structure in markup open selection dialog box for | public Properties[] listMarkupFiles(Properties intgInfo)<br>{<br>Properties[] retValues=null;<br>ArrayList markupFilesList = new ArrayList();<br>//Collect markup files list from the vault<br>String szDocumentId = intgInfo.getProperty("docid");<br>String |

<table>
<tr>
<td>integration.<br><br>Return Value:<br>An array of Java Properties object is returned to populate the tree structure in eReview Markup Open dialog box. Each of the properties object holds following key and value pairs.<br>title = document name to be displayed in tree leaf nodes.<br>path = the absolute document path will be loaded when selected.<br>size = to display the size of the file next to the leaf name.<br>docid = to identify the document uniquely in the integration.<br>isFolder = is set to false always..</td>
<td>sz_DocumentPath=intgInfo.getProperty("DOC_PATH");<br>//Lets say there are markup files saved into a specific folder.<br>String szMkpFileSavePath = "C:\\MyDocPath\\Markups";<br>File dir=new File(szMkpFileSavePath);<br>File[] listFiles= dir.listFiles();<br>if(listFiles != null)<br>{<br>File curent_file= null;<br>String szMkpTitle = null;<br>Properties currentMkpProperties = null;<br>String szMkpPath = "";<br>String szMkpId = "";<br>String szMkpFileSize = "";<br>for(int i= 0; i< listFiles.length; i++){<br>currentMkpProperties = new Properties();<br>curent_file= listFiles[i];<br>szMkpTitle = curent_file.getName();<br>//If you set this to "dummy", then the file will be read when selected from the list at the applet end.<br>szMkpPath = curent_file.getAbsolutePath();<br>szMkpId = String.valueOf(i);<br>szMkpFileSize = String.valueOf(curent_file.length());<br>currentMkpProperties.setProperty("title", szMkpTitle);<br>currentMkpProperties.setProperty("path", szMkpPath );<br>currentMkpProperties.setProperty("size", szMkpFileSize);<br>currentMkpProperties.setProperty("docid", szMkpId);<br><br>currentMkpProperties.setProperty("isFolder", "false");<br>markupFilesList.add(currentMkpProperties);<br>}<br>}<br>int totalNodeCount = markupFilesList.size();<br>retValues = new Properties[totalNodeCount];<br>for ( int nodeCount = 0; nodeCount < totalNodeCount; nodeCount++){<br>retValues[nodeCount] = (Properties)markupFilesList.get(nodeCount);<br>}<br><br>return retValues;<br>}</td>
</tr>
<tr>
<td>saveMarkupFile(Properties intgInfo)<br><br>Interface Argument:<br>intgInfo- A Java Properties object</td>
<td>public Properties saveMarkupFile(Properties intgInfo)<br>{<br>Properties retProp=new Properties();</td>
</tr>
</table>

| | |
|---|---|
| that holds all the parameters required for save markup integration.<br><br>**Return Value:**<br>A Java Properties object is returned with two parameters.<br>The “mkp_saved” is set to “true”/”false” to decide the success or failure. | String szDocumentId = intgInfo.getProperty("docid");<br>String szMkpFilePath = intgInfo.getProperty("mkpfilepath");<br>String szMkpName = intgInfo.getProperty("mkpname");<br>//Do your markup check in operation here to place the file into the vault or repository<br>retProp.setProperty("mkp_saved","true");<br>return retProp;<br>} |
| savePlaybackFile(Properties intgInfo)<br><br>**Interface Argument:**<br>intgInfo- A Java Properties object that holds all the parameters required for the save meeting information for playback integration.<br><br>**Return Value:**<br>A Java Properties object is returned with two parameters.<br>The “savedplayback” is set to “true”/”false” to decide the success or failure. | public Properties savePlaybackFile(Properties intgInfo)<br>{<br>Properties retProp=new Properties();<br>String szPlaybackPath = "C:\\MyMeetingPath\\Meeting1\\Meeting1.xml";<br>String szSysPBPath=intgInfo.getProperty("xmlfilepath");<br>try{<br>copyFile(szSysPBPath,szPlaybackPath);<br>}<br>catch(Exception e){<br>}<br>retProp.setProperty("savedplayback","true");<br>return retProp;<br>} |

## 5.2 Set Extension Class Parameters

The extension class and the additional parameters required to implement integration process in the extension class are provided as applet tag parameters. A sample applet tag for eReview integration through extension class is specified as follows:

```
<applet height="100%" width="100%" name="eReviewApplet"
code="chat.applet.CHttpChatApplet.class"  codebase="http://localhost:8080/ereview"
width=95% height=95%  >
<param name='console' value='+ALL'>
<param name='view.common.props.0' value='ereview.server.path-separator:%5C'>
<param name='uniqueid'
value='ASP.NET_SessionId=axm3ac55zamciueiy2un5b55'>
<param name='scriptable' value='true'>
<param name='remoteIp' value='127_0_0_1'>
<param name='IntegrateDMSCGI' value='CIntegrateServlet'>
<param name='ShowOpenButton' value='false'>
<param name='StringData' value='true'>
<param name='SERVER_TYPE' value='NOTUNIX'>
<param name='code' value='chat.applet.CHttpChatApplet.class'>
<param name='progressbar' value='true'>
<param name='boxmessage' value='Loading eReview Applet...'>
<param name='archive' value='EReviewLite.jar'>
<param name='codebase' value='.'>
```

```
<param name='server_url' value='EReviewServlet'>
<param name='multimediaDMSCGI' value='CMultimediaServlet'>
<param name='sock_emul_url' value='CSockEmulServlet'>
<param name='image' value='images/logo.gif'>
<param name='boxbgcolor' value='255,255,255'>
<param name='progresscolor' value='10,0,255'>
<param name='boxfgcolor' value='192,192,192'>
<param name='type' value='application/x-java-applet;version=1.5'>
<param name='Attendee_ID' value='Ashok'>
<param name='docname' value='MyDoc1.txt'>
<param name='sessionID'
value='ASP.NET_SessionId=axm3ac55zamciueiy2un5b55'>
<param name='view.common.props.2' value='ereview.viewonly.window-size:800'>
<param name='CLOSEWINDOW' value='true'>
<param name='load_GUI_conf_path' value='profile_user_Beginner View Only.xml'>
<param name='integratemode' value='true'>
<param name='integrator' value='allegria.oem.sample.SampleImplementation'>
<param name='integratename' value='allegria.oem.sample.SampleImplementation'>
<param name='integrateParams'
value='integratemode;Attendee_Name;Meeting_ID;Attendee_ID;StringData;MKPEDI
T;remoteIp;uniqueid;integratename;docname;sessionID;integrator;Markups_Path;Do
cument_Path'>
<param name='docPath' value='C%3A%5C%5CMyDocPath%5C%5CMyDoc1.txt'>
<param name='MAYSCRIPT' value='true'>
<param name='EREVIEW' value='VC'>
<param name='BRIEFCASE' value='false'>
<param name='JOIN' value='true'>
<param name='MODE' value='VIEW_ONLY'>
<param name='MEETING_ID' value='15-UE'>
<param name='Meeting_ID' value='15-UE'>
<param name='AUTODETECT' value='true'>
<param name='USE_LOCAL_DLL' value='FALSE'>
<param name='MKPEDIT' value='false'>
<param name='USERID' value='Ashok'>
<param name='USERNAME' value='Ashok'>
<param name='READONLY_ACCESS' value='false'>
<param name='Attendee_Name' value='Ashok'>
<param name='Document_Path' value='C:\MyDocPath\MyDoc1.txt'>
</applet>
```

The above html tag is generated through following few URLs. In this case allegria.oem.sample.SampleImplementation is an extension class extended from IOEMIntegrate calss. This sample is provided with ereview.war deployment and accessible using following URL.

**Sample URL to View Document:**

http://localhost:8080/ereview/CIntegrateServlet?REQUEST-TYPE=VIEWINTEGRATEDOC&Attendee_ID=Ashok&MKPEDIT=false&docname=MyDoc1.txt&integratename=allegria.oem.sample.SampleImplementation

**Sample URL to View and Annotate Document:**

http://localhost:8080/ereview/CIntegrateServlet?REQUEST-TYPE=VIEWINTEGRATEDOC&Attendee_ID=Ashok&MKPEDIT=true&docname=MyDoc1.txt&integratename=allegria.oem.sample.SampleImplementation

**Sample URL to View with specified profile:**

The profile profile_user_BeginnerViewOnly.xml is available in ereview\web-inf\profiles folder.

http://localhost:8080/ereview/CIntegrateServlet?REQUEST-TYPE=VIEWINTEGRATEDOC&Attendee_ID=Ashok&MKPEDIT=false&&Attendee_Name=Ashokena&Profile=profile_user_BeginnerViewOnly.xml&docname=MyDoc1.txt&integratename=allegria.oem.sample.SampleImplementation

A sample integration also provided with the ereview.war deployment to modify code in a jsp page and get it compiled on the fly to implement a quick integration. Use the following URL to access the sample page to check out eReview integration framework. The extended class always invokes the ereviewintegrator.jsp for each of the extension class events. Developers can modify the code in the jsp page to incorporate integration related business logic to access different files.

.

http://localhost:8080/ereview/testjspintegration.jsp

The HTTP request for integration carries three external parameters to implement the integration. Those are jspdocuser, jsppage and jspdocname. For every IOEMIntegrate implemented interfaces these three parameters are provided through the applet session. In each interface implementation the testjspintegration.jsp is invoked to receive data for the integration process. Developer set his business rule based on the parameters provided by the applet in the jsp class to implement his integration. Sample URL to view a document is provided in following URL.

http://localhost:8080/ereview/CIntegrateServlet?REQUEST-TYPE=VIEWINTEGRATEDOC&MKPEDIT=false&Attendee_ID=Ashok&jspdocUser=Ashok&jsppage=ereviewintegrator.jsp&jspdocname=document.tiff&integratename=allegria.oem.jspdocs.JSPIntegrator

# Chapter 6: Meeting Integration

eReview meetings require four mandatory parameters to start/join a meeting session. Those four mandatory parameters are “Meeting Title”, “Meeting ID”, “Attendee Name”, “Attendee ID” and “Join Mode”.

In case “Meeting ID”, “Attendee ID” are not known or provided through the integration process, eReview Servlet process auto generates from specified “Meeting Title”, “Attendee Name”. There are two ways to start an eReview meeting session as described in following section.

## 6.1 Applet Tags In HTML page

A HTML page with following minimum applet tag could start eReview meeting instantly when loaded from eReview application server

```
<applet height="100%" width="100%" name="eReviewApplet"
code="chat.applet.CHttpChatApplet.class"  codebase="http://localhost:8080/ereview"
width=95% height=95%  >
<param name='archive' value='EReviewNew.jar'>
<param name='code' value='chat.applet.CHttpChatApplet.class'>
<param name='type' value='application/x-java-applet;version=1.5'>
<param name='console' value='+ALL'>
<param name='scriptable' value='true'>
<param name='boxbgcolor' value='255,255,255'>
<param name='boxfgcolor' value='192,192,192'>
<param name='progresscolor' value='10,0,255'>
<param name='boxmessage' value='Loading eReview Applet...'>
<param name='progressbar' value='true'>
<param name='image' value='images/logo.gif'>
<param name='sock_emul_url' value='CSockEmulServlet'>
<param name='server_url' value='EReviewServlet'>
<param name='multimediaDMSCGI' value='CMultimediaServlet'>
<param name='codebase' value='.'>
<param name='MAYSCRIPT' value='true'>
<param name='EREVIEW' value='VC'>
<param name='SERVER_TYPE' value='NOTUNIX'>
<param name='CLOSEWINDOW' value='true'>
<param name='BRIEFCASE' value='true'>
<param name='MODE' value='EREVIEW'>
<param name='AGENDALINES' value='1'>
<param name='AGENDA' value=''>
<param name='JOIN' value='false'>
<param name='USERFOLDER' value='C:\\eReviewTomcat\\jakarta-tomcat-
5.0.28\\webapps\\ereview\\WEB-INF\\samples\\'>
<param name='USE_LOCAL_DLL' value='FALSE'>
<param name='MEETINGTITLE' value='MyMeetingTitle'>
<param name='MEETING_ID' value='MyMeetingID'>
<param name='MtPass' value='SUD'>
<param name='MKPEDIT' value='true'>
<param name='USERNAME' value='MyName'>
<param name='USERID' value='MyID'>
</applet>
```

6.1.1 Set “**JOIN**” to true if you want to join the same meeting as a different user. In this case two other parameters need to be changed to identify a new participant in the meeting. These parameters are “**USERNAME**” and “**USERID**”.

6.1.2 Set “**USERFOLDER**” to a specific folder on the server hard drive to read folders, sub folders and documents through document open dialog box at the applet end. In this case “**BRIEFCASE**” value has to be set as true.

6.1.3 Set “MtPass” to protect the meeting from external access. If this is set then every joining participant required to pass the password to join the on going meeting.

6.1.4 Set “**MKPEDIT**” to true if you want markup tools to be enabled to create annotations on the documents.

6.1.5 The “**USERNAME**” and “**USERID**” are required to set a unique Chairperson or participant in the meeting. In case of starting a meeting these parameters are used as chairperson name and id.

6.1.6 Set “**USE_LOCAL_DLLS**” to true to enable eReview accelerator dlls. Setting this to true downloads eReview viewer dlls to the client side hard drive and views the document locally.

## 6.2 Request with HTTP Header Parameters

As an alternative to 6.1 there are few HTTP request and response implementation in eReview server to generate HTML pages based on the request headed parameters. HTTP requested header keys are described in the following table:

Send request to http://localhost:8080/ereview/CIntegrateServlet

| Header Key | Default Value | Description |
|---|---|---|
| Attendee_ID | User identification Code | This identifies user applet session |
| Attendee_Name | User Name to Identify Markups | Markup layer created is associated with this name. |
| Meeting_ID | Unique ID | A meeting identifier to hold the meeting on the server. |
| MEETINGTITLE | Meeting Title | Sent Any name that needs to be displayed on web browser title |
| Document_Path | Document Path to List documents from server | This is the document path on the server. For a specific meeting a document path could be specified to load documents. |
| DocumentTitle | Title of the Document | To Display a different title of the document other than the filename itself |
| MKPEDIT | False/true | This could be set to true or false to enable markup tools or disable markup tools. |
| MtPassword | xxxxx | Use a password to protect the meeting from unauthorized access |

**Start Meeting URL:**
CIntegrateServlet?REQUEST-TYPE=START_MEETING&Attendee_ID=MyID&Attendee_Name=MyName&Meeting_ID=MyMeetingID&Meeting_Title=MyMeetingTitle&Document_Path=C:\\MyDocs\\ &MtPassword=SUD

**Join Meeting URL:**
CIntegrateServlet?REQUEST-TYPE= JOIN_MEETING&Attendee_ID=MyID&Attendee_Name=MyName&Meeting_ID=MyMeetingID &Meeting_Title=MyMeetingTitle&Document_Path=C:\\MyDocs\\ &MtPassword=SUD

**Start Playback Meeting URL:**
CIntegrateServlet?REQUEST-TYPE=PLAYBACK&XML_LOG_PATH=C:\\eReviewTomcat\\jakarta-tomcat-5.0.28\\webapps\\ereview\\WEB-INF\\samples\\playback.xml

# Chapter 7: Meeting Integration Extension

Meeting Integration Extension classes are implemented in same way as it is explained in Chapter 5. eReview server side integration hooks are designed to extend and implement the checkout and check-in process to read and write documents into ECM vault. Please follow Chapter 5 to implement meeting integration extension for Meeting Integration Extension.

The sample integration is available in ereview.war deployment at following URL.
http://localhost:8080/ereview/testjspintegration.jsp

Here are two sample URLS that could be used to try Meeting integration:

**Sample URL to Start Meeting:**

http://localhost:8080/ereview/CIntegrateServlet?REQUEST-TYPE=STARTINTEGRATEMEETING&Meeting_ID=YourMeetingID&StringData=true&Attendee_Name=Kevin&Attendee_ID=Kevin&join=false&meetingfolder=C:\\MyDocPath&integratename=allegria.oem.sample.SampleImplementation

**Sample URL to Join Meeting:**

http://localhost:8080/ereview/CIntegrateServlet?REQUEST-TYPE=STARTINTEGRATEMEETING&Meeting_ID=YourMeetingID&StringData=true&Attendee_Name=Tony&Attendee_ID=Tony&join=true&meetingfolder=C:\\MyDocPath&integratename=allegria.oem.sample.SampleImplementation

# Chapter 8: Embedding eReview Applet

eReview applet is possible to use within a third-party applet as a viewer engine. Only programming required for this is to include the EReviewNew.jar in your <applet/> tag and instantiate eReview applet instance within your applet. Now it is easy to access eReview interface methods to perform different actions in eReview viewer as required.
Following tables explain most of the eReview public facing interfaces.

## 8.1 Instantiate/Start eReview meeting session from another Applet

### 8.1.1 Start eReview Starter Session

```
ereview.oem.applet.IEReviewStarter erStarter =
new EReviewStarter();
```

### 8.1.2 Instantiate eReview Viewer session

```
ereview.oem.applet.IEReviewApplet applet =
new ereview.oem.applet.EReviewApplet();
```

### 8.1.3 Set Online EReviewServlet URL

```
String szSvrURL = "http://localhost:8080/servlet/EReviewServlet";
applet.setServletUrl (szSvrURL);
```

### 8.1.4 Set Online CIntegrateServlet URL

```
String szIntSvrURL = "http://localhost:8080/servlet/CIntegrateServlet";
applet.setServletIntegrateUrl (szIntSvrURL);
```

### 8.1.5 Set Online Meeting ID

```
String szMeetingID = "MeetingID";
applet.setMeetingID (szMeetingID);
```

#### 8.1.6 Set Viewing Session User Identification

```
String szAttendeeID = “MyID”;
applet.setAttendeeID(szAttendeeID);
```

#### 8.1.7 Set Viewing Session Documents path.

```
String szDocRootPath =
“D:\eReviewProf\eReviewProf\webpages\ereview\samples”;
applet.setDocRootPath(szDocRootPath);
```

#### 8.1.8 Set Document location need to be opened as soon as Viewing session starts.

```
String szDocRootPath =
“D:\ eReviewProf\webpages\ereview\samples\viewonly\exploded.tif”;
applet.setDocPath(szDocPath);
```

### 8.2 Join eReview meeting session

#### 8.2.1 Start eReview Starter Session

```
ereview.oem.applet.IEReviewStarter erStarter =
new EReviewStarter();
```

#### 8.2.2 Instantiate eReview Viewer session

```
ereview.oem.applet.IEReviewApplet applet =
new ereview.oem.applet.EReviewApplet();
```

#### 8.2.3 Set Online EReviewServlet URL

```
String szSvrURL = http://localhost:8080/servlet/EReviewServlet;
applet.setServletUrl (szSvrURL);
```

#### 8.2.4 Set Online CIntegrateServlet URL

```
String szIntSvrURL = http://localhost:8080/servlet/CIntegrateServlet;
applet.setServletIntegrateUrl (szIntSvrURL);
```

#### 8.2.5 Set Online Meeting ID

```
String szMeetingID = “MeetingID”;
applet.setMeetingID (szMeetingID);
```

#### 8.2.6 Set Viewing Session User Identification

```
String szAttendeeID = “MyID”;
applet.setAttendeeID(szAttendeeID + "joined");
```

#### 8.2.7 Set Viewing Session Documents path

String szDocRootPath = “D:\…\ereview\samples”;
applet.setDocRootPath(szDocRootPath);

#### 8.2.8 Set Document location need to be opened as soon as Viewing session starts.

String szDocRootPath = “D:\….\ereview\samples\viewonly\exploded.tif”;
applet.setDocPath(szDocPath);

## 8.3 Open document from server in eReview session

#### 8.3.1 The user has to create eReview Document instance by using

ereview.oem.applet.IEReviewDocument doc = new EReviewDocument();

#### 8.3.2 Document Path has to be set using

doc.setDocumentPath(szOpenDocPath);

#### 8.3.3 Using the following, user sets the title of the document

doc.setTitle(szOpenDocTitle)

#### 8.3.4 The document type (this type dictates the file format of the document) is set by using

doc.setDocumentAutodetectType(szOpenDocType);

#### 8.3.5 The document is opened using viewer Instance and return document id for future reference.

String szOEMDocId = erInst.openDocument(doc);

## 8.4 Close Opened Documents in eReview applet session

#### 8.4.1 Get all document instances from viewer instance.

IEReviewDocument[] docs = erInst.getAllDocuments();

#### 8.4.2 Match the documents for your szOEMDocId for which markups need to be saved.

#### 8.4.3 Search docs array to match the szOEMDocId. Let’s say matching id is I.

#### 8.4.4 Get the matching document object for matching number I.

IEReviewDocument curdoc = docs[I];

#### 8.4.5 Close offline document

erInst.closeDocument(curdoc);

## 8.5 Create markups and save created markups into server side directory

### 8.5.1 Get all document instances from viewer instance.

```
IEReviewDocument[] docs = erInst.getAllDocuments();
```

### 8.5.2 Match the documents for your szOEMDocId for which markups need to be saved.

### 8.5.3 Search docs array to match the szOEMDocId.

Let's say matching id is I.

### 8.5.4 Get the matching document object for matching number I.

```
IEReviewDocument curdoc = docs[I];
```

### 8.5.5 Save markup files at a given location with a given file name.

```
File fDestFile = new File(szSaveMkpPath);
String path = erInst.saveActiveMarkupFile(curdoc, fDestFile);
```

## 8.6 Save markups created on current document

### 8.6.1 Select Markup File Name on the server.

```
String szMkpFilePath = "D:\….\ereview\samples\sample.mkp";
```

### 8.6.2 Search docs array to match the szOEMDocId.

Let's say matching id is I.

### 8.6.3 Get the matching document object for matching number I.

```
IEReviewDocument curdoc = docs[I];
```

### 8.6.4 Save markup file on selected document.

```
File fMkpFile = new File(szMkpFilePath);
erInst.saveActiveMarkupFile (fMkpFile, curdoc);
```

## 8.7 Load/Unload markups on documents opened in eReview

### 8.7.1 Get all document instances from viewer instance.

```
IEReviewDocument[] docs = erInst.getAllDocuments();
```

### 8.7.2 Match the documents for your szOEMDocId for which markups need to be saved.

**8.7.3 Search docs array to match the szOEMDocId.**
Let's say matching id is I.

**8.7.4 Get the matching document object for matching number I.**

IEReviewDocument curdoc = docs[I];

**8.7.5 Load markup file on selected document.**

File fMkpFile = new File(szLoadMkpPath);
erInst.loadMarkupFile(fMkpFile, curdoc);

**8.7.6 Unload markup file on selected document.**

erInst.UnloadMarkupFile(szLoadMkpPath, curdoc);

## 8.8 Print current or selected document

**8.8.1 To print the current document,**

erInst.printCurrentDocument ();

**8.8.2 To print the selected documents, follow the steps:**

**8.8.3 Get all document instances from viewer instance.**

IEReviewDocument[] docs = erInst.getAllDocuments();

**8.8.4 Match the documents for your szOEMDocId for which markups need to be saved.**

Search docs array to match the szOEMDocId. Let's say matching id is I.

**8.8.5 Get the matching document object for matching number I.**

IEReviewDocument curdoc = docs[I];

**8.8.6 To print the selected document, use:**

erInst.printSelectedDocuments(curdoc);

## 8.9 Close Collaboration Session

**In order to leave or end the meeting session use following API:**

erInst.close()

# Chapter 9: Java Beans and ActiveX Bridge

eReview applet functionality is accessible from a windows ActiveX component. Most of the ActiveX component can load eReview ActiveX bridge and start interaction with java beans developed as a eReview wrapper class.

Our solutions utilize a Java virtual machine (VM) that is loaded as a dynamically linked library (DLL) into the client application. It's important to note that only one instance of the VM per client is loaded. This allows an application to "connect" multiple ActiveX controls to each other, and know that that their underlying Java Bean instances are running in a shared VM.The registry file (reg), generated from eReview Bean, and contains information about the ActiveX control, including the location of its type library (tlb) and jar files. This file is loaded into the registry so that ActiveX client applications can access the control. The type library contains information about the COM interface(s) supported by the control. Applications use this file to provide API information to the control user. The Java methods and parameter names are preserved, making the controls easier to use. Although this file is binary, one can examine its contents with the OLE-COM Object Viewer that comes with Visual C++. This file is similar to any other type library. There is nothing Java-specific about it. Inside the type library there are declarations for the public Interfaces of the eReview Java Bean, as well as the interface for objects that will receive events generated by your eReview Java Bean.This information is generated automatically from the BeanInfo class of the eReview Java Bean. You don't need the registry and type library files to use the control. You need them only to check the methods. Users of the eReview ActiveX control won't necessarily know they're using a Java Bean. It will appear along with the
other ActiveX controls and have almost all the same characteristics found in controls written in other languages like Visual C++. When an eReview ActiveX control is loaded into an application, the registry information instructs the system to load the Java virtual machine as a dynamically-linked library (assuming it hasn't already been loaded) and create an instance of the eReview Java Bean class. Subsequent method calls and events are passed between the application and the control via some specially generated adapter classes. When deployed, eReview ActiveX control will require the following items to be installed on the customer's machine:

- **The Java Runtime Environment (JRE), version 3.0 or higher**
- **The eReview.ocx ActiveX control**
- **The eReview client jar file (EReviewNew.jar)**

The viewer may be controlled entirely by the presentation layer without any direct viewer access to the document management system and without demanding its own window. The design of the user interface may be completely controllable by the client program. All own eReview graphical tools elements (Toolbars, Status bar) may be switched off. All GUI functions (zooming, rotation, and tiling) are callable as ActiveX methods. All methods use only String or Void type parameters making the integration easier. All major events are propagated to the application.To summarize, the eReview as a multi format viewer may be incorporated into the presentation layer of any Windows application that uses ActiveX controls. In the same time, eReview as an Applet can be used for integration into any browser based solution in J2EE environment.

More information on this is provided in a separate documentation.

# Chapter 10: Java Script Command Tokens

Before executing Java Script Command Tokens it is necessary to access eReview Applet object. eReview applet has exposed an interface to receive the input from external web application framework. The exposed interface in turn triggers an event to process the command instruction provided to it. This integration process follows the steps described in following section.

Integrate eReview using Java Script is described in following section:

10.1 Use <applet/> tag to hold eReview applet instance in a HTML page. Specify correct code base to load EReviewNew.jar in your web application. At the same code base ereview.war is deployed with eReview servlets. Refer the <applet/> tag syntax specified in section 4.1 of Chapter 4.

10.2 Access eReview Applet object instance within Java Script function.

**Sample Code:**
If the name of the applet tag is specified as follows
name="eReviewApplet"
Then the sample code to access the eReview Applet Object would be as follows.

```
<script>
    var eReviewAppletObject = document.eReviewApplet;
</script>
```

10.3 Invoke execeReviewAPI() interfacing method of the eReview Applet object to send the action command along with the required parameters. Save the return value for further integration actions if necessary. The interface has two arguments to take. The first argument is to decide the command action where as the other argument is dedicated to take parameters to execute the action.

**Sample Code:**

In this example there are two keys used to provide parameter information such as File and Title. File represents the absolute document path and Title represents the Title to be displayed on the MDI child window displaying the document.

Task- Open document C:\\DocumentFolder\\document.tiff
Action Command: OpenDocument
Command Parameters: File= C:\\DocumentFolder\\document.tiff
The sample code to execute this command would be as follows:

```
<script>
  var eReviewAppletObject = document.eReviewApplet;
  var szCommand = “OpenDocument”;
  var szParameter=”File=C:\\DocumentFolder\\document.tif,Title=MyDocument”;
  eReviewAppletObject.execeReviewAPI(szCommand, szParameter);
</script>
```

10.4 The ereview.war deployment has some more java script implemented sample code in following URL. Following are some of the Java Script based APIs implemented in eReview. We have developed sets of frames to access from client side. User can also create his own command line and execute using applet methods:
**http://localhost:8080/ereview/ReviewDocs/MainFrame.html**

10.4.1 **OpenDocument :** This command is used to open documents from server or client side using eReview applet session.

Action Command: OpenDocument
Command Parameter: File=C:\DocumentFolder\\document.tif, Title=MyDocument

10.4.2 **Get Document ID:** This command is used to get the unique ID of the document. Provide the document path as parameter. Collect the Document ID to provide in markup load, unload and save operations. Lets say you

received DocId=MrX_11232132432 use them in your later API commands as required.

Action Command: GetDocumentID
Command Parameter: DocumentPath=C:\DocumentFolder\\document.tif

10.4.3 **Load Markup:** We use this command to load an existing markup on the current document in focus or the DocId specified in the parameter. Provide the markup path as a mandatory parameter.

Action Command: LoadMarkupFile
Command Parameter:
DocId=MrX_11232132432,MarkupFile=C:\TEMP\Markups\MyMarkup.mkp,
FileLocation=Server

10.4.4 **Save Markup:** This command is for saving a markup on the current document in focus or the DocId specified in the parameter. Provide the markup path as a mandatory parameter.

Action Command: SaveMarkupFile
Command Parameter:
DocId=MrX_11232132432,MarkupFile=C:\TEMP\Markups\MyMarkup.mkp,
FileLocation=Server

10.4.5 **Unload Markup:** To unload an existing markup from the current document in focus or the DocId specified in the parameter. Provide the markup path as a mandatory parameter.
Action Command: UnloadMarkupFile
Command Parameter:
DocId=MrX_11232132432,MarkupFile=C:\TEMP\Markups\markup.mkp

10.4.6 **Create New Markup Layer:** This command is used to create a new Markup layer. Provide the MarkupFile, Layer and UserId as mandatory parameters.

Action Command: CreateLayer
Command Parameter:
DocId=MrX_11232132432,,MarkupFile=C:\TEMP\Markups\markup.mkp,
,Layer=LayerForMrX,UserId=MrX

10.4.7 **Activate Markup Layer:** If we want to activate a particular layer in our eReview Applet then we use this command. Provide the MarkupFile, Layer and UserId as mandatory parameters.

Action Command: ActivateLayer
Command Parameter:
DocId=MrX_11232132432,,MarkupFile=C:\TEMP\Markups\markup.mkp,
,Layer=LayerForMrX,UserId=MrX

10.4.8 **Delete Markup Layer:** If we want to delete a particular layer from our eReview Applet then we use this command. Provide the MarkupFile, Layer and UserId as mandatory parameters.

Action Command: DeleteLayer
Command Parameter:
DocId=MrX_11232132432,,MarkupFile=C:\TEMP\Markups\markup.mkp,
,Layer=LayerForMrX,UserId=MrX

10.4.9 **CloseDocument:** If we want to close some opened documents then we use this command. Provide the DocumentId as mandatory parameters.

Action Command: DeleteLayer
Command Parameter: DocumentId=Dave_11232132432 for single document and
DocumentId=(Dave_43256758,Dave_2312321312) for multiple documents.

10.4.10 **Close All Documents :** This command is used to close all the opened documents. Here we don't need any parameter. We just click the button to execute the request.

10.4.11 **Save Drawing As Raster Image:** When we want to save any drawing in the form of a raster image (gif, Jpeg etc) we use this command. Use this command to load an existing markup on a document. We need to give the markup path as parameter.

Action Command: SaveDrawingAsRaster
Command Parameter: OutputFile= C\\Exploded_DaveWithMarkup.jpg, Type=JPEG, IncludeMarkups= Yes

10.4.12 **Apply Stamps:** If we want to apply some stamps on a document then we use this. We give the name, size, color and position as parameters.

Action Command: StampDocument
Command Parameter: Stamp=Not Approved , RelXPos=50, RelYPos=60,FgColor= (255,0,0), BgColor= (255,255,255),Font=Arial , Size= 12, Bold=true , Italic=false, Underline=false

10.4.13 **Switch Applet Instance:** In some implementation it is required to place eReview applet in different tab frames of a single HTML page. In order to execute zoom and pan operations through java script you need to switch eReview applet current instance.

Action Command: SwitchApplet
Command Parameter: AppletTagName

10.4.14 **Zoom In:** To zoom in an active document we use this command. No parameter is use here.

Action Command: ZoomIn
Command Parameter: None

10.4.15 **Zoom Out:** To zoom out an active document we use this command. No parameter is use here.

Action Command: ZoomOut
Command Parameter: None

10.4.16 **Rotate Right:** In order to rotate the current document we take help of this document. We don't send any parameter as a request.

Action Command: RotateRight
Command Parameter: None

10.4.17 **Rotate Left:** In order to rotate the current document we take help of this document. We don't send any parameter as a request.

Action Command: RotateLeft
Command Parameter: None

10.4.18 **Tile Document:** To tile more than one document in the eReview window we use it. There is no need of any parameter. We just click the button to execute the request.

Action Command: TileDocument
Command Parameter: None

10.4.19 **Reset:** After Zoom In/Out or rotate right/left when we want to reset the actions on a document we use this command. We don't need to send any parameter as a request.

Action Command: Reset
Command Parameter: None

# Chapter 11: Configuration Parameters

**11.1 Preventing the user-profiles dialog box from popping up as soon as the applet loads**

<property key="ereview.skip.profiles" value="true" />

**11.2 Enabling auto loading of markups with the parent document, add the following PARAM value to the applet parameters**

<property key="ereview.markups.autoload" value="true" />

**11.3 Preventing the "save markup" dialog box from appearing. The markups now would get savedwith a pre-defined name (markup.mkp) and date and timestamp.**

<property key= "ereview.show.savemkp" value= "false"/>

**11.4 Opening DLL based documents**

<property key="ereview.dll.dir" value="C:\jakarta-tomcat-5.0.28\webapps\ereview\web-inf\evef"/>

<property key="ereview.servlets.dir" value="C:\jakarta-tomcat-5.0.28\webapps\ereview\web-inf\lib\eReviewServlet.jar"/>

If these two are correct, then DLL based documents should open smoothly.

### 11.5 Enabling fast PDF viewing

<property key= “ereview.fastviewpdf.mode” value= “true”/>

### 11.6 Enabling fast Tiff Viewing

<property key= “ereview.fastview.mode” value= “true”/>

# Chapter 12: Customized Tool Buttons

eReview viewer allows to add external action into its own tool buttons. Define external actions in a java program. Compile the class file and bundle in a jar file. Define a new button in eReview UI configuration XML file as sown in the following figure. Specify the jar file name in <jar_name/> tag.

**Java Source**

```
Package com.web4.ereview.custom;
import chat.applet.CHttpChatApplet;
public class CustomAction implements
java.awt.event.ActionListener
{
private CHttpChatApplet applet;
public CustomAction ()
{
        super();
}
public CustomAction (Object apl)
{
        super();
        applet = (CHttpChatApplet)apl;
}
public void
actionPerformed(java.awt.event.ActionEvent
e)
{
//Add your custom action here…


}
}
```

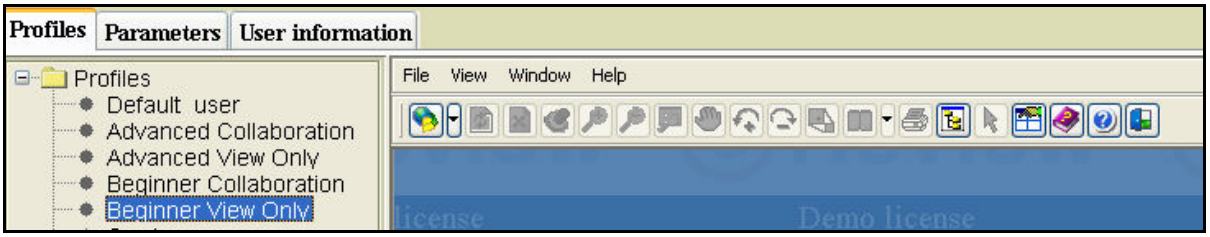
**eReview UI configuration XML**

```
<button>
   <name>CUSTOM_ACTION</name>
   <visibile>true</visibile>
   <tooltip>Custom Action</tooltip>
   <mnemonic_pos>1</mnemonic_pos>
   <hot_key />
   <start_x>96</start_x>
   <start_y>128</start_y>
   <width>16</width>
   <height>16</height>
<image_src>/images/toolbar5.gif</image_src>
   <is_toggle>false</is_toggle>
   <button_type>ACTION_TYPE</button_type>
   <jar_name>customaction.jar</jar_name>
<listener>com.web4.ereview.custom.CustomAct
ion</listener>
   <listener method />
</button>
```

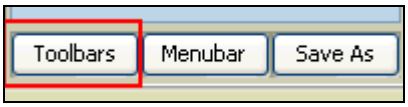
Add this to the xml profile file where you want to introduce the new button. The location of the file is C:\..\ereview\WEB-INF\profiles

12.1 Login to the admin module from the eReview interface.

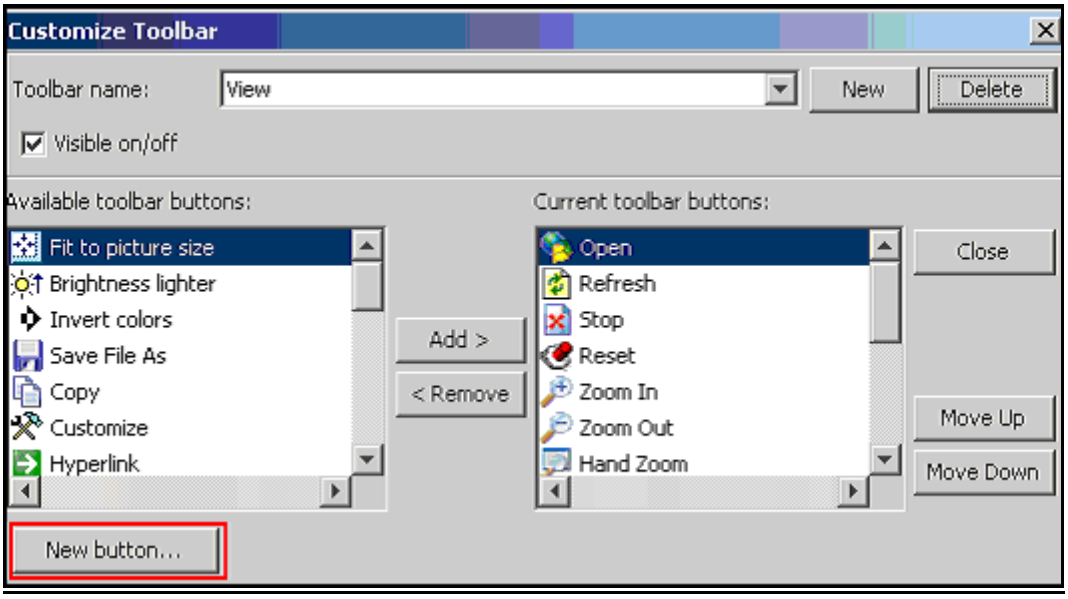
12.2 From the left panel, select the profile where you want to add the new button.



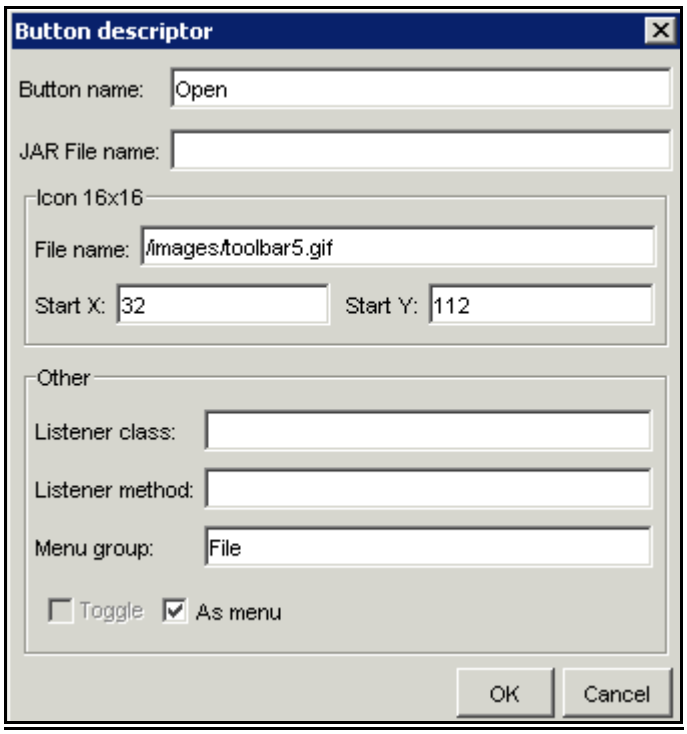
12.3 Next, click on the " Toolbars" tab at the bottom of the left panel .




12.4 To add a new button, click on the "new button" tab on the pop-up window.




12.5 You should see a "button descriptor" pop-up window .Enter the relevant details and click "OK" to add a new button to the interface.




# Chapter 13: Control User Profile

The eReview user interface is customizable entirely according to the requirements of the users. The toolbars, menubars and look-and-feel of the interface can be configured through a separate module called the eReview admin module. There are some user profiles that are shipped with eReview's default installation. The administrators having knowledge of user-profiles can edit and/or create user-profiles, depending on the use-cases in the actual deployment.Profiling helps scale the product interface according to the user level i.e. from novice to advanced.There are some default profiles that are shipped with eReview. These profiles can be tied to a particular user name. When the user logs in with that specific id .

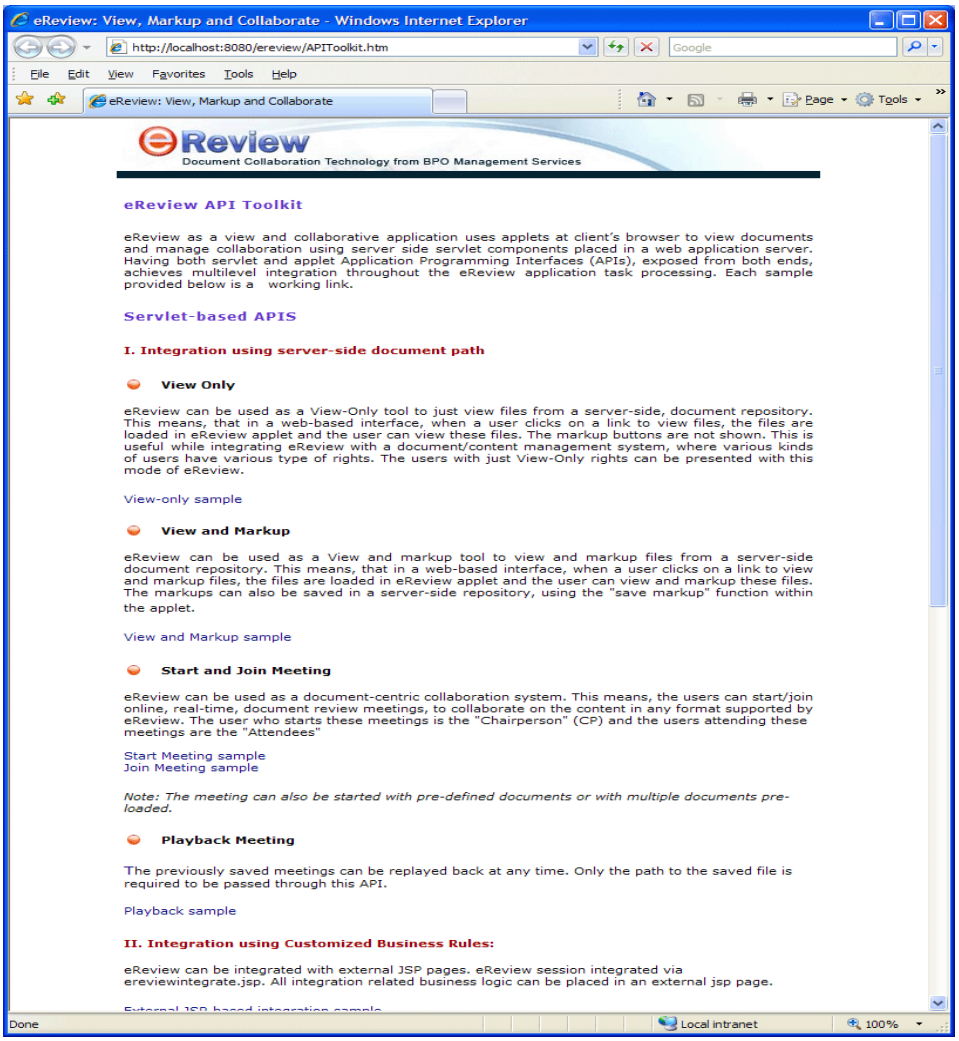
Click here to see different sample profiles of eReview

Sample URL to load sample profile "sampleProfile.xml :

*CIntegrateServlet?REQUEST-TYPE=VIEWINTEGRATEDOC&Attendee_ID=Ashok&MKPEDIT=false&docname=Exploded.tif&integratename=allegria.oem. integrate.CustomIntegrate&Profile=sampleProfile.xml*

# Chapter 14 : Check from APIToolkit.html

eReview deliverable war file (ereview.war) has few sample API implementations available in APIToolkit.html. The sample page has several links to click and check the implemented functionality.




Click here to try live APIToolKit.html from demo.ereviewonline.com.

**http://demo.ereviewonline.com/ereview/APIToolkit.htm**