PEARSON ALWAYS LEARNING

EQUELLA®

Scripting Guide (Advanced)

Version 6.4

Document History

Document No.	Reviewed		Finalised		Published	
1	22/05/2015		22/05/2015		22/05/2015	

May 2015 edition.

Property of:

Pearson
Level 1, 11 Elizabeth Street
Hobart, Australia, 7000

Contact documentation@equella.com for matters relating to documentation.

Table of Contents

Scripting overview

EQUELLA provides scripting to enable system administrators to exercise fine grain control over processes within EQUELLA.

This guide is intended for administrators familiar with JavaScript™, FreeMarker and HTML and describes the Advanced Scripting control and other script-related information.

While this guide describes advanced scripting, EQUELLA also provides basic script editors that are easy to use and provides sophisticated management of items, controls and security allowing the construction of scripts based on item metadata, user role or item status. Refer to the *EQUELLA Scripting Guide (Basic)* for more information on basic scripting.

Please note that this guide has been developed to best reflect the full capabilities of EQUELLA and as such may differ in appearance to your own installation.

Advanced Scripting control

The Advanced Scripting control requires advanced knowledge of scripting languages.

This section describes the **Advanced Scripting** wizard control that is available in the **Collection Definition Editor** within the Administration Console. Add an Advanced Scripting control to a new collection (or an existing one) to explore the features of the control. An example is shown in Figure 1.

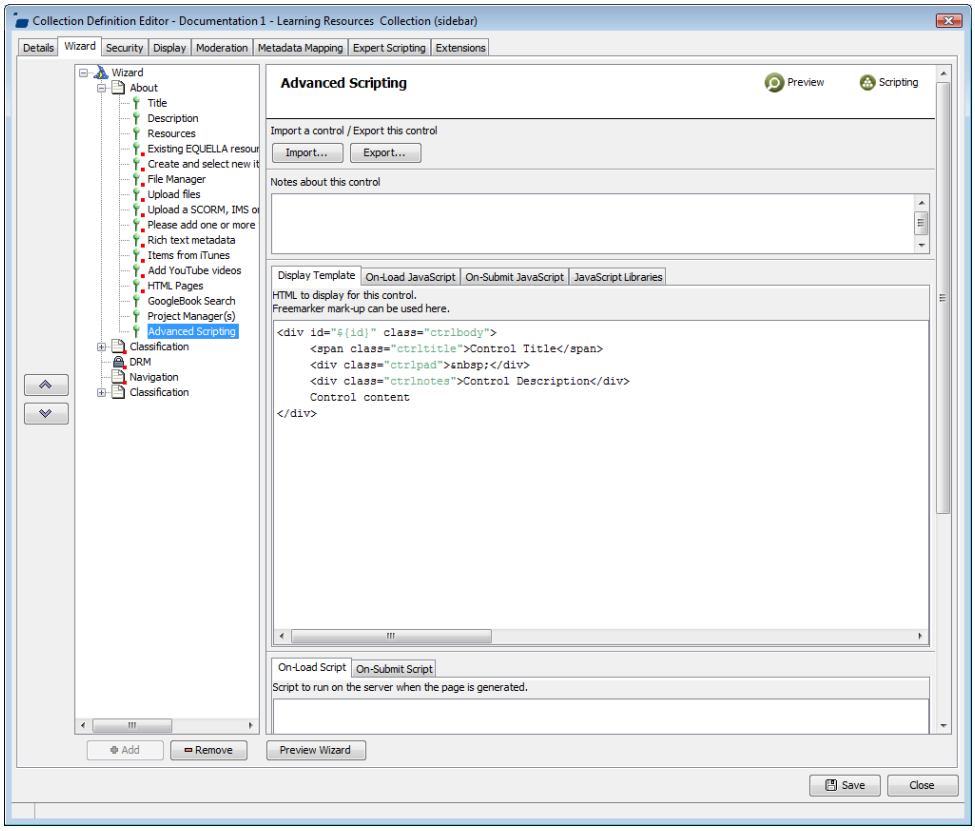

Figure 1 Collection Definition Editor—Advanced Scripting control

The Advanced Scripting control elements include:

- **Import...** button—select to import an advanced script configuration (*.**asc** file). Further information is provided in Advanced scripting control examples on page 10.
- **Export.**..button—select to export an advanced script configuration (*.**asc** file).
- **Notes** pane—add descriptive information about the control.

There are a number of different script panes on the control, separated into two different groups based on whether they are sent to a browser or not.

The tabs at the top (**Display Template**, **On-Load JavaScript**, **On-Submit JavaScript**) all use FreeMarker mark-up and are evaluated on the server before being sent to the user's browser. (*NOTE: Any JavaScript in these panes is **not** evaluated on the server; you do not have access to the script objects such as 'item' in this JavaScript (although the script objects **are** available in FreeMarker*).)

The tabs at the bottom (**On-Load Script**, **On-Submit Script**) use server-side JavaScript™ and should be where you perform most of your item, XML and page/control visibility manipulation.

The order that the scripts are evaluated is:

1. **On-Load Script** (on the server).
2. **On-Load JavaScript**, **On-Submit JavaScript**, **Display Template** (on the server. FreeMarker evaluation only, *not* the JavaScript).
3. **On-Load JavaScript** (by the user's browser).

When the user submits the page (e.g. via one of the wizard commands such as Next, Submit etc. or via a self-reloading control) the remaining scripts are executed in this order:

4. **On-Submit JavaScript** (by the user's browser)
5. **On-Submit Script** (on the server)

If the page containing the Advanced Scripting control is reloaded (e.g. because of a self-reloading control), the first three scripts/FreeMarker are re-evaluated.

If you need to pass custom information from the server panes to the client-side panes, you should store it in the **attributes** object in your **On-Load Script** and read the values stored in the **attributes** in the Freemarker templates.

The **attributes** object is an `XmlScriptType`, so it has the same methods as the **xml** object (see the Script API documentation).

FreeMarker basics

FreeMarker is an open-source template engine for rendering dynamic text, primarily HTML pages.

The Advanced Scripting control **Display Template** and the client-side JavaScript text fields accept FreeMarker mark-up. You do not have to use FreeMarker mark-up if you simply want to render some static text.

FreeMarker tags are like HTML tags, but with a '#' character in front of the tag name. Common FreeMarker tags are:

```
<#if test == true>Some true text</#if>
<#list someList as someItem>${someItem}</#list>
<#assign someNewVariable = someExpression />
```

To print the value of a variable, the variable needs to be enclosed by a `${}`.

For example:

```
<#if myVariable == true>${someVariableToPrint}</#if>
```

You can use comments in FreeMarker using `<#-- My comment -->`.

FreeMarker comments will not be rendered in the final HTML output.

A more comprehensive FreeMarker manual is available at:
http://fmpp.sourceforge.net/freemarker/index.html

JQuery basics

JQuery is an open-source JavaScript library used extensively by EQUELLA. You have access to the core JQuery library functions in all of the client-side panes. You also have access to functions in other JQuery extension libraries such as Pretty Photo. To use these functions, you will need to import the relevant library on the **JavaScript Libraries** tab of the **Advanced Scripting** control.

For a comprehensive introduction to JQuery visit this site:

http://docs.jquery.com/Tutorials

FreeMarker panes

The FreeMarker panes are the client-side panes, that is, the **Display Template**, the **On-Load JavaScript** and the **On-Submit JavaScript**.

All FreeMarker variables are referenced by enclosing the name of the variable in

`${}`, for example. `${prefix}`, unless you are within a FreeMarker tag such as

```
<#if>, e.g. <#if prefix == 'something'>
```

All regular script variables are available in the FreeMarker panes, such as `xml`, page and `user`. These additional variables are available to the Advanced Scripting control:

- **attributes**—this variable is available in all panes. It is an XML object (behaving exactly like the `xml` variable). You can use this object to pass information between panes. For example, you can do pre-processing JavaScript in the On-Load server-side script, push information into the `attributes` object, and then have this information available in the `attributes` object in the Freemarker panes.
- **prefix**—the unique prefix for this control. Any input tags you render should have a name which is prefixed by this variable, otherwise you will not be able to read it back on the server using the `request` script object. For example:

```
<input id= "${prefix}myInput" name="${prefix}myInput" type="text">
```

  - To retrieve this value on the server when the page is submitted, you would place this in your server-side **On-Submit Script**:

```
var submittedValue = request.get('myInput');
```

- **submitJavascript**—this will render JavaScript to submit the page. You would normally use it on a button. For example:

```
<input id="${prefix}mySubmit" name="${prefix}mySubmit" type="submit" value="Submit Me"
onclick="${submitJavascript}">
```

  - On the server, to detect if this submit button was pressed you would use code similar the following in your server-side **On-Submit Script**:

```
if ( request.get('mySubmit') != null )
{
    // Do something
}
```

- **previewUrlBase** —this is the relative path you would use to generate URLs for previewing attachments on the current item. For example:

```
<img src="${previewUrlBase}someImageAttachment.jpeg">
```

On-Load JavaScript

This script is executed by the browser each time the page containing the Advanced Scripting control is displayed.

If you use any FreeMarker code in the **On-Load JavaScript** pane it will be evaluated on the server before being sent to the user's browser.

On-Submit JavaScript

If you use any FreeMarker code in the **On-Submit JavaScript** pane it will be evaluated on the server before being sent to the user's browser.

Server-side script panes

All regular script variables are available in the server-side script panes, such as `xml`, page and user. The On Submit script has an additional `request` variable available to read submitted values from your mark-up. Only fields prefixed with the `${prefix}` variable in your **Display Template** will be available in the `request` object. When reading from the `request` object there is no need to specify the prefix. For example:

```
var myInput = request.get('myInput');
```

Or to detect if a custom submit button has been clicked:

```
if ( request.get('mySubmit') != null)
{
// Do something
}
```

Available script objects

The names of the available script objects are listed below. For a full description of all the available objects and their methods please refer to the Script API documentation.

Object name	Description	API Reference
attachments	Manipulates the attachments on the current item.	com.tle.common.scripting.objects AttachmentsScriptObject
attributes	Available in an Advanced Script Control context only. This object is used to pass information between scripts, without polluting the item xml.	com.tle.common.scripting.types XmlScriptType
catalogue	View and assign catalogues.	com.tle.core.payment.scripting.objects CatalogueScriptObject
ctrl	A reference to this Advanced Scripting control.	com.tle.web.wizard.scripting.objects ControlScriptObject
currentItem	Retrieve information about the current item.	com.tle.common.scripting.types ItemScriptType
data	Retrieve information from any available Taxonomy.	com.tle.core.taxonomy.scripting.objects TaxonomyServiceScriptObject
drm	Retrieve DRM information from the current item.	com.tle.web.scripting.advanced.objects

		DrmScriptObject
images	For image manipulation, such as resizing.	com.tle.web.scripting.advanced.objects ImagesScriptObject
items	Retrieve information about other items.	com.tle.common.scripting.objects ItemScriptObject
lang	Retrieve the current user's time-zone and language information.	com.tle.web.scripting.advanced.objects RegionalScriptObject
logger	For logging information to the Resource Center log files. Useful for debugging scripts.	com.tle.common.scripting.objects LoggingScriptObject
meta	Used for summary display templates. Enables you to add HTML meta tags to the page.	com.tle.web.discoverability.scripting.objects MetaScriptObject
metadata	Retrieve extracted metadata from attached files.	com.tle.core.metadata.scripting.objects MetadataScriptObject
mime	Get MIME-type information for files.	com.tle.web.scripting.advanced.objects MimeScriptObject
nav	For modifying package navigation nodes.	com.tle.web.scripting.advanced.objects NavigationScriptObject
newxml	Available only in New Version scripts. This represents the XML of the new version of the item and 'xml' represents the XML of the current version.	com.tle.common.scripting.types XmlScriptType
page	A reference to the current wizard page.	com.tle.web.wizard.scripting.objects PageScriptObject
request	Available in an Advanced Script Control context only. Used for retrieving submitted values from the page.	com.tle.web.controls.advancedscript.scripting.objects RequestMapScriptObject
staging	For manipulating files on disk within the current staging folder.	com.tle.common.scripting.objects FileScriptObject
status	The current status of the item.	One of these String values: draft, live, rejected, moderating, archived, suspended, deleted, review, personal
system	Available only in a	com.tle.common.scripting.objects

	contribution wizard and Advanced Script Control context only. Allows you to invoke helper applications, e.g. to extract data from video files.	SystemScriptObject
tier	View and assign pricing information.	com.tle.core.payment.scripting.objects PricingTierScriptObject
url	Available only in the URL Checking scheduled task.	com.tle.core.scripting.objects UrlsScriptObject
user	For retrieving details about the current user such as name and group/role membership details. This object can also be used to retrieve information about other users as well.	com.tle.common.scripting.objects UserScriptObject
utils	Useful functions that don't specifically fit into any other objects. These functions include searching for items and downloading data from external URLs.	com.tle.common.scripting.objects UtilsScriptObject
workflowstep	Available only during contribution wizards. This is the UUID of the current workflow task.	This is a String value
xml	Contains the item metadata for reading and writing.	com.tle.common.scripting.types XmlScriptType

Advanced scripting control examples

The **EQUELLA Integration Pack** contains a number of exported Advanced Scripting controls (*.**asc** files) that you can import and modify. The EQUELLA Integration Pack is available from the **Downloads** tab in the EQUELLA **Institution Manager**.

To access the Institution Manager

1. Enter the complete EQUELLA URL to your institution with *'/institutions.do?method=admin'* appended to the URL *(*e.g. *'http://equella.myinstitution.edu/logon.do'* would become *'http://equella.myinstitution.edu/institutions.do?method=admin'*).
2. Log in using the EQUELLA server administrator password.
3. Selecting the **Downloads** tab to display the **Downloads** page.
4. Select the **Download the Integration Pack** link shown in Figure 2, and download the **equella-integration-pack.zip** file.

Figure 2 EQUELLA Integration Pack

To import an Advanced Scripting control

1. Add an Advanced Scripting control to a collection's wizard using the Collection Definition Editor in the Administration Console.
2. Click the **Import...** button on the new control.
3. Select one of the sample *.**asc** files included in the Integration Pack.

Redirection servlet

EQUELLA includes a special URL redirection servlet which is useful to avoid cross-site scripting restrictions (particularly with AJAX functions). The servlet is available at '*http://YOUR_INSTITUTION_URL/p/geturl?url=ENCODED_URL'*. For example: '*http://myserver.edu.au/institution/p/geturl?url=http%3A%2F%2Fwww.google.com'.*

The servlet will follow any HTTP 302 redirects and return the contents of the final page with the same HTTP response code.

Example

If you wish retrieve the contents of an external URL within an **Advanced Scripting** control, you might place something like this in your **Display Template**:

```
<script type="text/javascript">
function showHtml(myExternalUrl)
{
    // Get the page from a remote server
    $.get('p/geturl', {'url': myExternalUrl},
            function(html)
            {
                    alert(html);
            }
            , 'html');
}
</script>

<input type="text" id="myUrl" value="http://www.yahoo.com/">
<input type="button" value="Get HTML!" onclick="showHtml($('#myUrl').val());">
```

Although this example has very little real-world use, it shows some slightly more advanced features of JQuery to access the redirection servlet. The `showHtml` function above uses the `$.get()` function, which is a core JQuery function to retrieve data from a URL.

- The first parameter is the URL to retrieve data from, in this case the redirection servlet at the relative path of 'p/geturl'.

- The second parameter is a list of query string parameters to this URL, the 'url' parameter on the redirection servlet in this case. These query string parameters will automatically be URL-encoded.
- The third parameter is the expected format of the response, i.e. HTML.

Contact Client Support

We are always happy to help.

If your organisation has a support agreement with EQUELLA then help is available at http://equella.custhelp.com/.