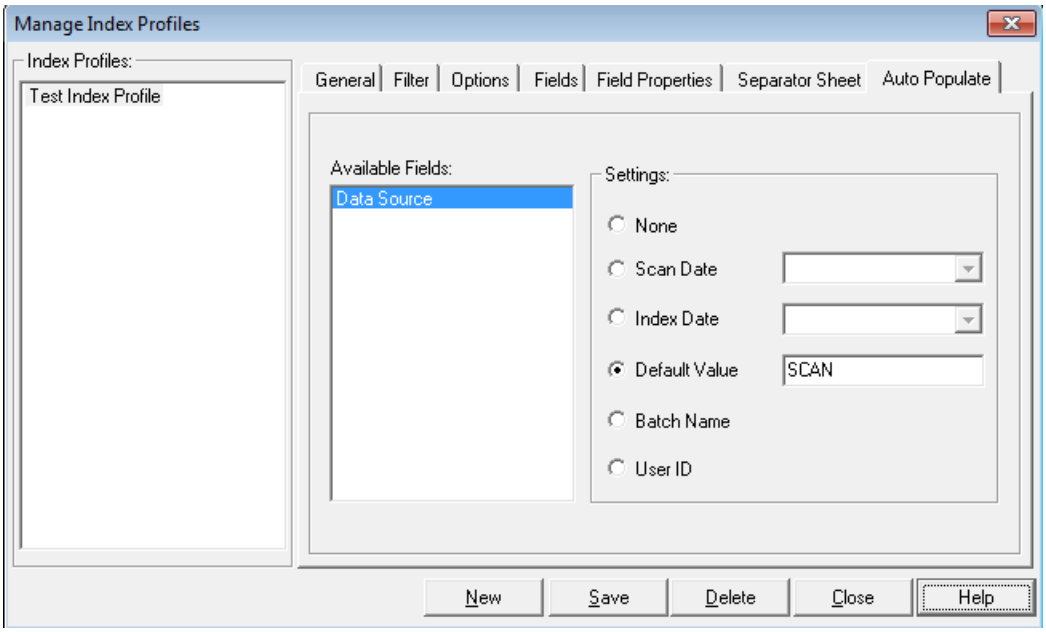
Creating and Debugging a Basic Script for WebCenter Enterprise Capture

Many customers using Oracle Document Capture 10g have customized their capture and index profiles using custom scripts written in VB. Since the release of WebCenter Enterprise Capture, these scripts are no longer compatible with the latest version of the software. WEC 11g now has a brand new API for customizations and scripts are written in JavaScript. For customers looking to upgrade, this means evaluating their scripts, deciding which are still needed and how they should be revamped using the 11g APIs.

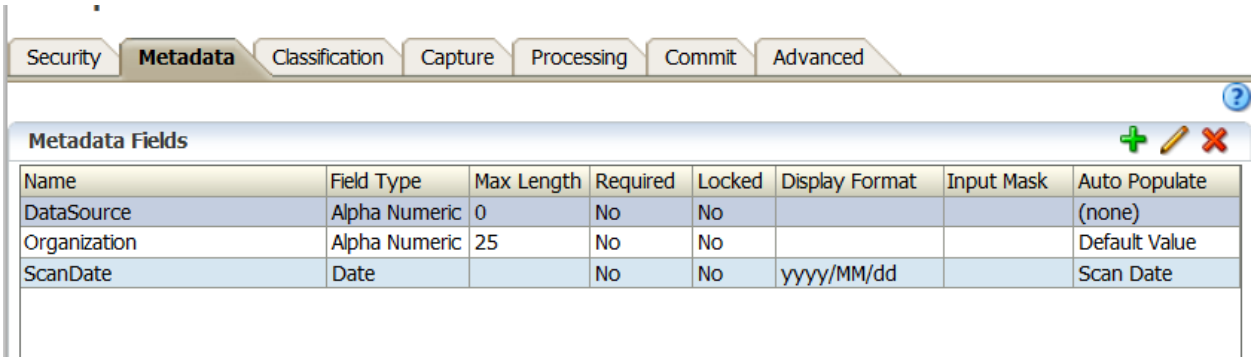
Unfortunately, since WEC 11g is still relatively new, there is not a lot of help out there beyond Oracle's documentation and a few sample scripts in Oracle Knowledge Base. This article walks you through how to create a simple script using the new APIs and using the Java console to debug it.

Goal: *Create a script to default a field value for a particular client profile.*

ODC 10g had the ability to default a field value based on the Index Profile used. This was very useful as often times a File Cabinet contains multiple scan and index profiles and a particular field's value may vary depending on the profile used.



WEC 11g does have the ability to auto populate values, however this is done only at the metadata level.



Name	Field Type	Max Length	Required	Locked	Display Format	Input Mask	Auto Populate
DataSource	Alpha Numeric	0	No	No			(none)
Organization	Alpha Numeric	25	No	No			Default Value
ScanDate	Date		No	No	yyyy/MM/dd		Scan Date

This article will demonstrate how to create a simple script that defaults a value for every document in the batch based on the client profile used. Oracle knowledge base provides a similar script that defaults a value when a document is selected (KB article 1639684.1).

```
var CODER_FIELD = "Coder";   // The name of the coder metadata field

function DocumentSelected(event) { // DocumentSelectedEvent

     var document = event.getDocument();

     var batch = document.getParentBatch();

     var fieldDef = batch.getWorkspace().getFieldDefinitions().findByName(CODER_FIELD);

     if (fieldDef != null) {

          var fieldId = fieldDef.getId();

          var fields = document.getFields();

          var field = fields.get(fieldId);


         // Set the field value to the logged-in user

          field.setValue(Capture.getCurrentUser().substring(0,5));


         // Save the document data

          document.persist();

     }

 }
```

The issue with this script is that since it fires on the `Document Selected` event, it requires that a user manually open every document in a batch in order for the value to be defaulted for that document. In some scenarios, this might be the only field that is captured by WEC, meaning there are no other fields to be indexed. This often happens for integrations with Forms Recognition; where only a few select values are captured at scan time, such as Organization, and the rest are extracted using Forms Recognition.

Writing the Script

Using the Oracle provided script as a basis; we create a new js file for our script. The script shown here will default a field called “DataSource” with a value of “SCAN”.

Begin by setting the global variables for the script, the name of the field and the default value for that field.

```
var DATA_SOURCE_FIELD = "DataSource";   // The name of the Data Source metadata field
var DATA_SOURCE_VALUE = "SCAN";      //Default value for the Data Source field
```

Looking through the available client events, the `Batch Selected` event will provide the needed functionality. Looking through the API, the `batch` object can be obtained from the `event` parameter. From the `batch` object, we can use the `getDocuments()` method to retrieve the documents in the batch.

```
var batch = event.getBatch();
var batchDocsVector = batch.getDocuments();
```

The `getDocuments()` method returns a `CaptureDocuments` object which is actually a Vector of `CaptureDocuments`. After some trial and error, I found that the simplest way to manipulate the documents is to convert the vector into an array.

```
var batchDocs = batchDocsVector.toArray();
```

From there, just as in the Oracle sample script, get the `fieldDefinition` and the `fieldId`. Next, loop through the documents in the array. For each document, get the desired field and set it to the default value, just as shown in the sample script.

```
var fieldDef =
batch.getWorkspace().getFieldDefinitions().findByName(DATA_SOURCE_FIELD);

   if (fieldDef != null) {
        var fieldId = fieldDef.getId();

        for(var i=0; i< batchDocs.length; i++)
        {
                //get the Document
                var doc = batchDocs[i];

                //get the field
                var fields = doc.getFields();
                var field = fields.get(fieldId);

                // Set the field value to the default value
                field.setValue(DATA_SOURCE_VALUE);

                // Save the document data
                doc.persist();

        }
   }
```

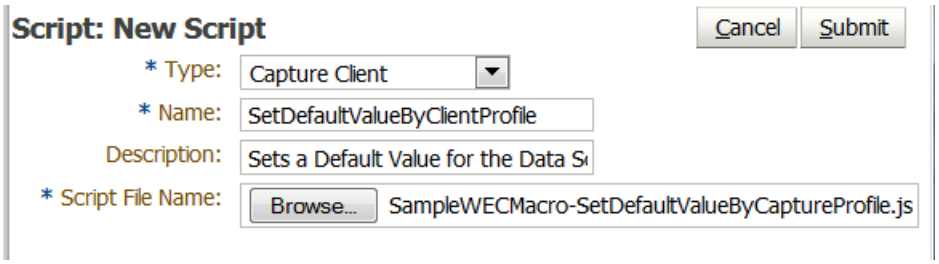
Putting it all together, the script looks like this:

```
var DATA_SOURCE_FIELD = "DataSource";   // The name of the Data Source metadata field
var DATA_SOURCE_VALUE = "SCAN";      //Default value for the Data Source field

 //This functions sets the default value for all documents in the batch when the batch is selected
 function BatchSelected(event) { // BatchSelectedEvent

   var batch = event.getBatch();
   var batchDocsVector = batch.getDocuments();
   var batchDocs = batchDocsVector.toArray();

   var fieldDef = batch.getWorkspace().getFieldDefinitions().findByName(DATA_SOURCE_FIELD);

   if (fieldDef != null) {
        var fieldId = fieldDef.getId();

        for(var i=0; i< batchDocs.length; i++)
        {
                //get the Document
                var doc = batchDocs[i];

                //get the field
                var fields = doc.getFields();
                var field = fields.get(fieldId);

                // Set the field value to the default value
                field.setValue(DATA_SOURCE_VALUE);

                // Save the document data
                doc.persist();

        }
   }

 }
```

Configuring the Workspace

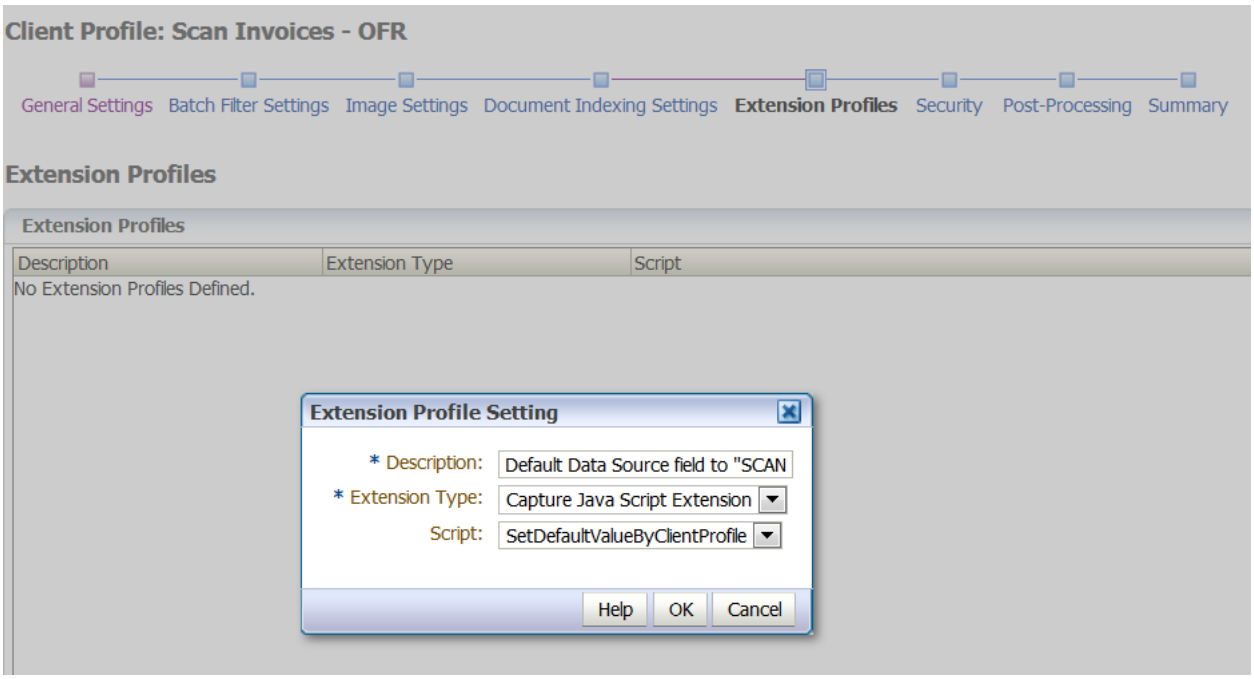
Next, configure the script in the workspace. Log into the Capture Console. In the desired workspace, create a new script in the Advanced tab.

Type: Capture Client
Name: <choose a name>
Description: <enter a description>
Script File Name: select the js file you created



After saving the script, the next step is to assign it to a client profile. In the Capture tab, open an existing client profile, or create a new one. The Profile Type can be either "2 - Capture and Index" or "3 – Index Only". On the Extension Profiles train stop, add the script you just created.

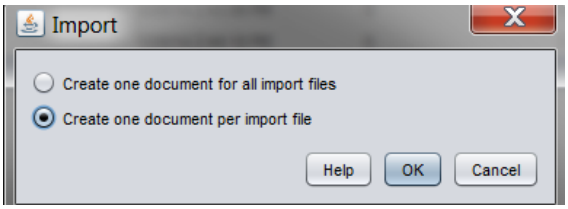
Description: <enter a description>
Extension Type: Capture Java Script Extension
Script: <select the script you just created>

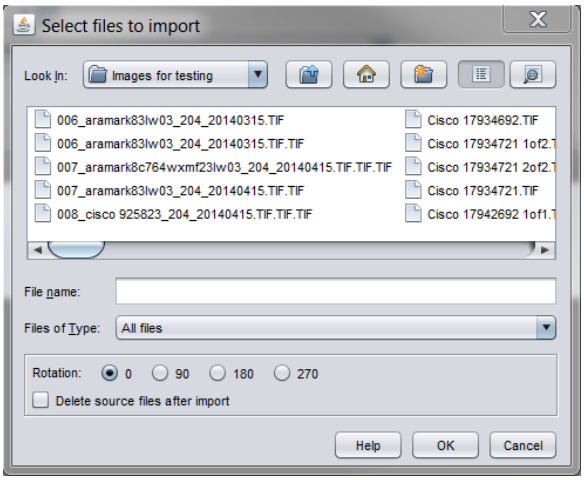


Be sure to save the changes. It's now time to test the script.

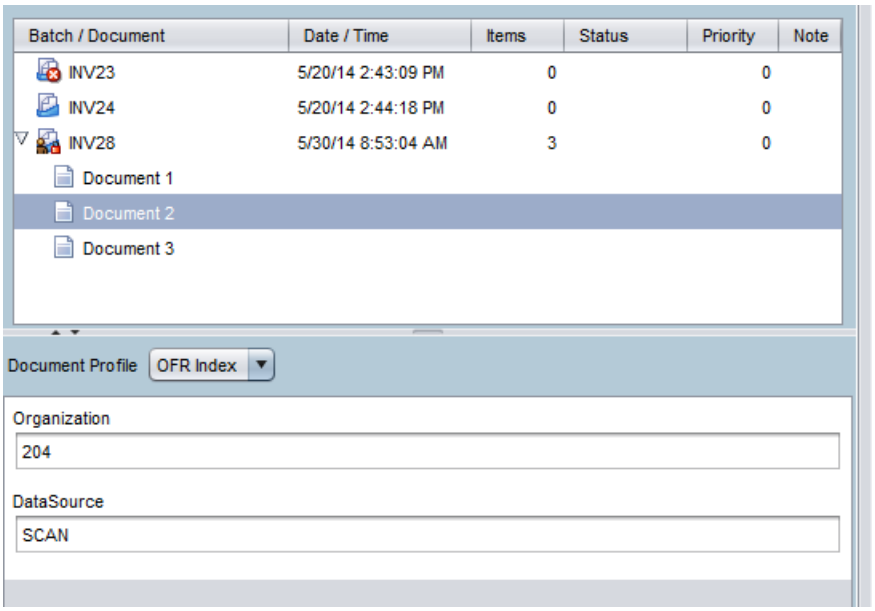
Testing the Script

Log into the Capture Console. Create a batch and open it for indexing using the client profile you configured in the section.

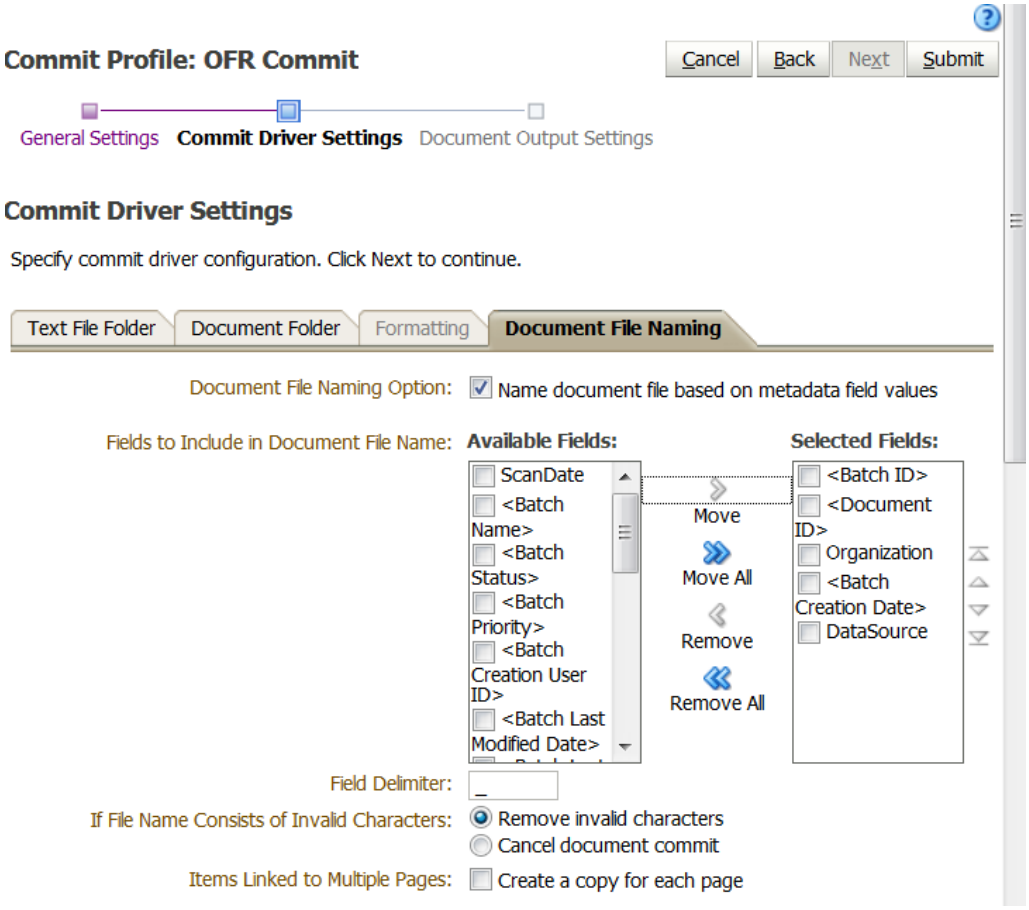




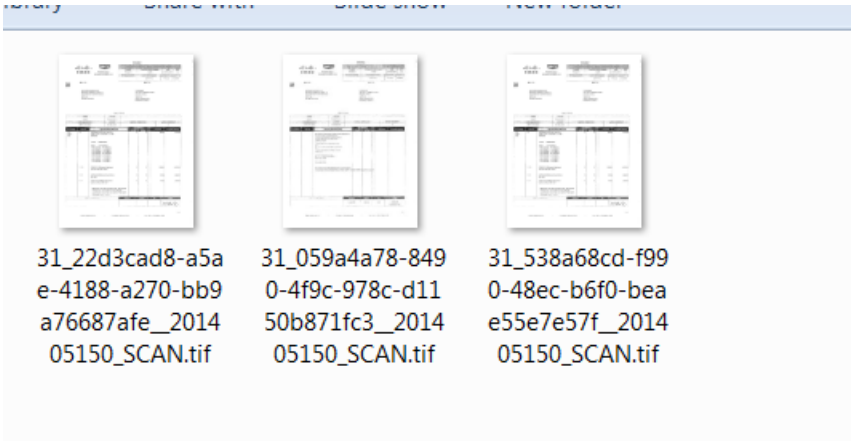
Since the batch is automatically selected after the scan/import is completed, the `BatchSelected` event will fire automatically after the scan/import is done. You can see that the script has completed successfully by looking at the metadata assigned to a document in the batch.



To thoroughly test that the script is working correctly, first configure the commit profile to include the new field with the exported metadata. In the case of exporting to WFR, this will be part of the file name, so below the DataSource field is added to the Document File Name.



After configuring the export in the workspace, log back into the Capture Console. Scan or import another document using the same client profile. After scanning, immediately release the batch without looking at the documents individually. Browse to the location where the committed documents are placed. The file names of the images should all have “SCAN” as one of the elements.

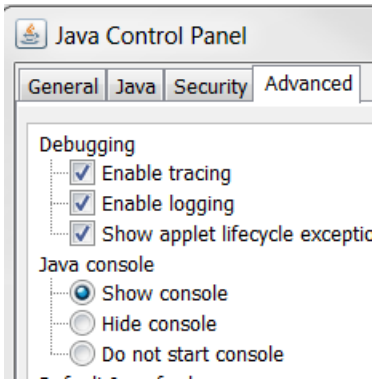


Debugging with the Java Console

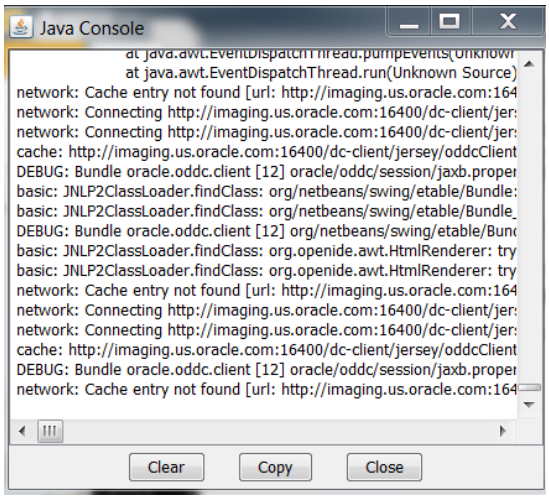
When writing scripts, it is useful to be able to trace the code as it is being run to verify that it is being executed as expected. WEC 11g scripts provide the ability to do this by writing to the java console. To add tracing to your script, simply call the `println` method.

```
println("SampleWECMacro: batchDocs size: " + batchDocs.length);
        for(var i=0; i< batchDocs.length; i++)
        {
                println("SampleWECMacro: Getting document " + i);
```

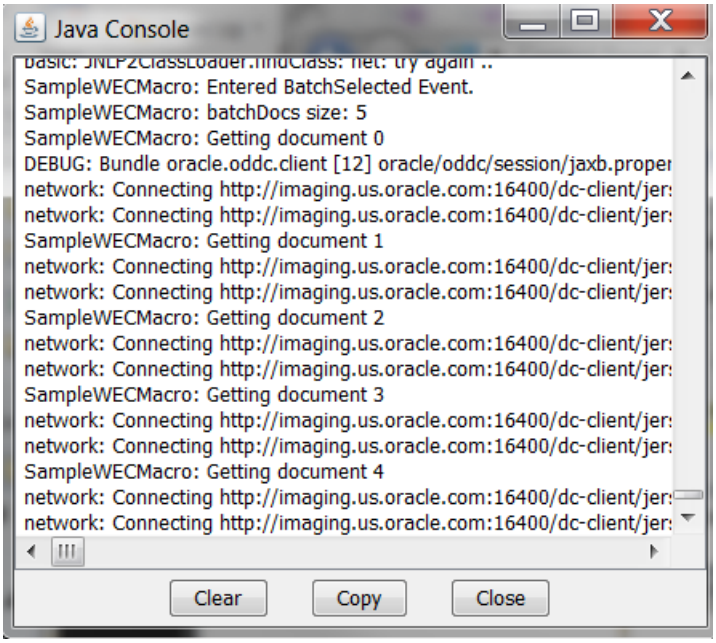
To see the output, you must enable the Java Console on the machine you are running the Capture Client from. On Windows 7, open the Java menu from the Control Panel. In the Advanced tab, Enable tracing and logging under the Debugging section and choose to Show console under the Java console section.



To enable the console, you must close the browser completely and reopen it. The next time you open the Capture Console, the Java Console will pop open automatically.



Scan a document again. This time, you will see the tracing lines from the script in the Java Console output.



This can be an invaluable tool for creating and debugging new Capture scripts. While the APIs are fairly straightforward, it is always useful to be able to trace through what is actually happening and being executed, especially when methods don't function exactly as expected.