



Ephesoft Transact Developer's Guide

Version: 2023.1.00

Date: 2023-08-18

© 2023 Kofax. All rights reserved.

Kofax is a trademark of Kofax, Inc., registered in the U.S. and/or other countries. All other trademarks are the property of their respective owners. No part of this publication may be reproduced, stored, or transmitted in any form without the prior written permission of Kofax.

Table of Contents

Preface	6
Disclaimer.....	6
Training.....	6
Getting help with Kofax products.....	6
Chapter 1: Custom plugins	8
Create custom plugins.....	8
Prerequisites.....	8
Create a custom plugin project.....	9
Add Transact libraries.....	9
Create a new Java interface.....	9
Create a Java class.....	10
Create folder structure for XML resources.....	11
Create XML files.....	11
Compile the custom plugin project.....	14
Prepare custom plugin for import.....	15
Import custom plugins.....	15
Add plugins to a batch class.....	16
Remove custom plugins.....	16
Chapter 2: External applications	18
Application security.....	19
Add an external application.....	19
Access the application.....	19
Transact AI Table Rule Builder.....	19
Chapter 3: Regular expressions	29
Predefined character classes of regular expressions.....	29
Quantifiers.....	30
Capturing groups.....	30
Backreferences.....	30
Capturing groups and character classes with quantifiers.....	31
Boundary matchers.....	31
Grouping constructs.....	31
Regex patterns in Transact.....	33
Usage of 'pattern' field in 'document Index Field Details'.....	33
Usage of multi-word in key pattern for k-v extraction.....	33

How not to capture certain values for key pattern and value pattern.....	34
Usage of multi-word capture in Table Extraction, which is different than Value Pattern in K-V Extraction.....	34
Chapter 4: Scripting resources.....	35
Scripting functionality.....	35
Batch Instance Group feature.....	35
Force Review feature.....	35
ScriptAddNewTable.java.....	36
ScriptAutomaticValidation.java.....	36
ScriptDocumentAssembler.java.....	37
ScriptExport.java.....	37
ScriptExtraction.java.....	37
ScriptFieldValueChange.java.....	38
FunctionKey.java.....	38
ScriptPageProcessing.java.....	39
ScriptValidation.java.....	39
Application Level script.....	39
Create the Application Level script file.....	40
Configuration.....	40
Execution.....	41
Dependency.....	41
Troubleshooting.....	41
Batch Instance Group.....	41
Batch.xml and XSD schema.....	42
Batch-level fields.....	42
Document fields.....	47
Document-level fields.....	48
Page fields.....	57
Page-level fields.....	61
Email metadata in the batch.xml schema.....	62
Case studies for batch.xml.....	62
Client-side scripting.....	63
Error causes for default scripts.....	65
Enable logging for custom scripts.....	66
JDOM script configuration.....	67
Sample scripts to compare IScript and JDOM.....	67
Implement .zip functionality to older scripts.....	69
Testing scripts.....	70

- Trigger field value change script for table data fields.....71
- Chapter 5: Transact Web Services API..... 73**
- Transact Web Services optimized for workflow engines..... 73
- Web service definitions and code samples..... 75
 - Authentication for web services..... 75
 - Batch class management web services..... 76
 - Batch instance management web services.....86
 - Batch instance processing web services..... 89
 - Image processing web services.....119
 - Reporting web services.....124
- Web services requests.....126

Preface

The *Ephesoft Transact Developer's Guide* is designed for developers that are looking to add additional functionality or customize their solution beyond the standard installation.

This guide includes the following topics:

- [Custom plugins](#)
- [External applications](#)
- [Regular expressions](#)
- [Scripting resources](#)
- [Transact Web Services API](#)

Disclaimer

Any customization related to this *Ephesoft Transact Developer's Guide* is considered outside the scope of a standard Transact deployment and is not covered under our support agreement. If you need assistance in creating, implementing, or troubleshooting a custom solution, contact Professional Services.


Training

Kofax offers both classroom and computer-based training that will help you make the most of your Ephesoft Transact solution. Visit the [Kofax Education Portal](#) for details about the available training options and schedules.

Getting help with Kofax products

The [Kofax Knowledge Portal](#) repository contains articles that are updated on a regular basis to keep you informed about Kofax products. We encourage you to use the Knowledge Portal to obtain answers to your product questions.

To access the Kofax Knowledge Portal, go to <https://knowledge.kofax.com>.

 The Kofax Knowledge Portal is optimized for use with Google Chrome, Mozilla Firefox, or Microsoft Edge.

The Kofax Knowledge Portal provides:

- Powerful search capabilities to help you quickly locate the information you need.
Type your search terms or phrase into the **Search** box, and then click the search icon.
- Product information, configuration details and documentation, including release news.
To locate articles, go to the Knowledge Portal home page and select the applicable Solution Family for your product, or click the View All Products button.


From the Knowledge Portal home page, you can:

- Access the Kofax Community (for all customers).
On the Resources menu, click the **Community** link.
- Access the Kofax Customer Portal (for eligible customers).
Go to the [Support Portal Information](#) page and click **Log in to the Customer Portal**.
- Access the Kofax Partner Portal (for eligible partners).
Go to the [Support Portal Information](#) page and click **Log in to the Partner Portal**.
- Access Kofax support commitments, lifecycle policies, electronic fulfillment details, and self-service tools.
Go to the [Support Details](#) page and select the appropriate article.

Chapter 1

Custom plugins

This chapter provides resources for developers looking to create and manage custom plugins. Custom plugins can be used to alter the default workflow of Ephesoft Transact to meet a specific use case. They can be Kofax-developed or user-developed, but they are maintained by the user.

 Before proceeding, review the [Disclaimer](#).

Create custom plugins

To create custom plugins, follow these general steps:

1. Fulfill prerequisites by installing necessary programs.
2. Create a new project in Eclipse.
3. Add Transact libraries.
4. Create a new Java interface.
5. Create a Java class.
6. Create a folder structure for XML resources.
7. Create XML files to be used as resources for the plugin.
8. Compile the project.
9. Import the plugin into Transact.

Prerequisites

To create a custom plugin for Ephesoft Transact, you will need the following installed on your system:

- [Java JDK](#)
- [Apache Maven](#)
- [Eclipse](#)
- [Maven](#)

To install Maven, perform the following steps:

1. Open Eclipse.
2. Go to **Help > Eclipse Marketplace**.
3. Search for **Maven**.
4. Click **Install** on **Maven Integration for Eclipse**.
5. Follow the installation steps as prompted.

6. After the installation is complete, go to **Window > Preferences**.
Maven is listed in the left panel.

Create a custom plugin project

Follow these steps to create a custom plugin project in Eclipse. For more information about using Eclipse and Maven, see the documentation provided with those products.

1. Open Eclipse.
2. Select **File > New**.
3. Select **Maven Project**.
4. Select **Use default Workspace location**.
5. Select the quickstart Maven archetype. The Artifact ID is **maven-archetype-quickstart**.
6. Enter the Group ID and Artifact ID.


The Group ID is the package structure for your project. The Artifact ID will be the name of your project, and the final source folder of your package. For more information about the Java specification for package names, refer to the Java SE specifications on the Oracle website.

- Group ID: com.ephesoft.customplugin
- Artifact ID: ephesoft-custom-plugin

Your Eclipse project workspace is now set up for your project.

Add Transact libraries

Follow these steps to add the Transact libraries to your Eclipse project.

 Custom script class files should not be added to any custom JAR files. Adding a custom script class file into any JAR file will cause issues, as the default location for the custom script class file is `<Transact_folder>\JavaAppServer\temp\DynamicCodeCompiler`.

1. Open your project folder. Right-click on **ephesoft-custom-plugin**.
2. Select **Build Path > Configure Build Path**.
3. Select the **Libraries** tab.
4. Click **Add External Jars**.
5. Navigate to `<Transact_folder>\Application\WEB-INF\lib`.
6. Select all JAR files.
7. Click **Open**.
8. Click **OK**.

Create a new Java interface

Follow these steps to create a new Java interface for your custom plugin.

1. Open your package. Right-click **com.ephesoft.customplugin.ephesoft.custom_plugin**.
2. Select **New > Interface**.
3. Set the **Name** to **CustomPlugin**.

4. Click **Finish**.
5. Replace the file contents with the following code:

```
package com.ephesoft.customplugin.ephesoft_custom_plugin;
import com.ephesoft.dcma.core.DCMAException;
import com.ephesoft.dcma.da.id.BatchInstanceID;
public interface CustomPlugin
{
    void helloWorld(final BatchInstanceID batchInstanceID, final String
pluginWorkflow) throws DCMAException;
}
```

Create a Java class

Follow these steps to create a Java class to implement the CustomPlugin interface.

1. Open your package. Right-click **com.ephesoft.customplugin.ephesoft_custom_plugin**.
2. Click **New > Class**.
3. Set the **Name** to **CustomPluginImpl** and click **Finish**.
4. Replace the file contents with the following code:

```
package com.ephesoft.customplugin.ephesoft_custom_plugin;
import org.springframework.util.Assert;
import com.ephesoft.dcma.core.DCMAException;
import com.ephesoft.dcma.core.annotation.PostProcess;
import com.ephesoft.dcma.core.annotation.PreProcess;
import com.ephesoft.dcma.core.component.ICommonConstants;
import com.ephesoft.dcma.da.id.BatchInstanceID;
import com.ephesoft.dcma.da.service.BatchClassPluginConfigService;
import com.ephesoft.dcma.util.BackUpFileService;
public class CustomPluginImpl implements CustomPlugin, ICommonConstants {
    private BatchClassPluginConfigService batchClassPluginConfigService;
    @PreProcess
    public void preProcess(final BatchInstanceID batchInstanceID, String
pluginWorkflow) {
        Assert.notNull(batchInstanceID);
        BackUpFileService.backUpBatch(batchInstanceID.getID());
    }
    @PostProcess
    public void postProcess(final BatchInstanceID batchInstanceID, String
pluginWorkflow) {
        Assert.notNull(batchInstanceID);
    }
    public void helloWorld(BatchInstanceID batchInstanceID, String pluginWorkflow)
throws DCMAException {
        //TODO Auto-generated method stub
        String name = "";
        String propertyName = "app.name";
        name =
batchClassPluginConfigService.getPluginPropertiesForBatch(batchInstanceID.getID(),
"EPHESOFT_CUSTOM_PLUGIN").get(propertyName);
        System.out.println("**** Ephesoft Custom Plugin: Hello " + name + " " +
batchInstanceID.getID() + " ****");
    }
    public BatchClassPluginConfigService getBatchClassPluginService() {
        return batchClassPluginConfigService;
    }
    public void setBatchClassPluginConfigService(BatchClassPluginConfigService
batchClassPluginConfigService) {
        this.batchClassPluginConfigService = batchClassPluginConfigService;
    }
}
```

Create folder structure for XML resources

Follow these steps to create a folder structure for your XML resources.

1. Right-click your project in the **Project Explorer**.
2. Click **New > Source Folder**.
3. Set the Folder Name to `src/main/resources`.
4. Click **Finish**.
5. Create a `META-INF` folder:
 - a. In the project folder you just created, right-click `src/main/resources`.
 - b. Select **New > Folder**.
 - c. Set the **Folder Name** to `META-INF`.
 - d. Click **Finish**.
6. Create a subfolder under `META-INF`.
 - a. Right-click the `META-INF` folder.
 - b. Select **New > Folder**.
 - c. Set the **Folder Name** to "ephesoft-custom-plugin" (or the name of your project, if different).
 - d. Click **Finish**.

Create XML files

The following XML files provide resources that are used by the custom plugin:

- `<name of project>-plugin.xml` (such as `ephesoft-custom-plugin.xml`)
- `applicationContext-<name of project>-plugin.xml` (such as `applicationContext-ephesoft-custom-plugin.xml`)
- `applicationContext.xml`
- `pom.xml`

The `pom.xml` file already exists. The other three XML files need to be created. You then need to add or replace the content of the files with the information provided in this section. Follow these steps:

1. Create the `ephesoft-custom-plugin.xml`, `applicationContext-<name of project>-plugin.xml`, and `applicationContext.xml` files as follows:
 - a. Right-click `src/main/resources`.
 - b. Select **New > XML File**.

 If **XML File** is not available, click **Other** and search for **XML File**.

- c. Set **Name** to the name of the file you are creating:
 - `<name of project>-plugin.xml` (such as `ephesoft-custom-plugin.xml`)

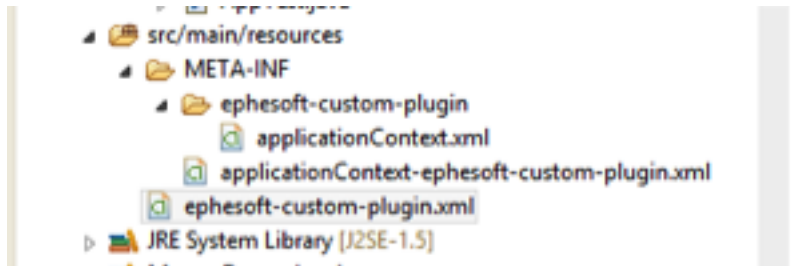
- applicationContext-<name of project>-plugin.xml (such as applicationContext-ephesoft-custom-plugin.xml)
- applicationContext.xml

d. Click **Finish**.

The file is created.

e. Repeat these steps to create all three files.

When you are finished, the folder structure looks similar to the following:



2. Open each file and add the contents in the following sections. Set the configurable elements if indicated.

For pom.xml, locate it in the project, open it, and replace the contents with the code provided in the applicable section below.

<name of project>-plugin.xml

Add the following content to the XML file you created:

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
  <jar-name>ephesoft-custom-plugin.jar</jar-name>
  <plugin-name>EPHESOFT_CUSTOM_PLUGIN</plugin-name>
  <plugin-desc>Ephesoft Custom Plugin</plugin-desc>
  <plugin-workflow-name>CUSTOM_PLUGIN</plugin-workflow-name>
  <plugin-service-instance>CustomPlugin</plugin-service-instance>
  <method-name>helloWorld</method-name>
  <is-scripting>FALSE</is-scripting>
  <back-up-file-name>EphesoftCustomPlugin</back-up-file-name>
  <script-name>N/A</script-name>
  <application-context-path>applicationContext-ephesoft-custom-plugin.xml</application-
context-path>
  <plugin-properties>
    <plugin-property>
      <name>app.name</name>
      <type>STRING</type>
      <description>Name</description>
      <is-mandatory>FALSE</is-mandatory>
      <is-multivalue>FALSE</is-multivalue>
    </plugin-property>
  </plugin-properties>
  <dependencies>
</dependencies>
</plugin>
```

The following elements are configurable. Change them as needed:

Element	Description
jar-name	Name of the JAR file containing your plugin code. This should be <name of the project>.
plugin-name	Name of the plugin to display in the workflow. The CustomPluginImpl.java class uses the name EPHESOFT_CUSTOM_PLUGIN.
plugin-service-instance	Name of the bean identifier to be referenced in the applicationContext.xml file.
method-name	Name of the method to call first when the plugin executes in the workflow.
plugin-properties	Defines the properties of the plugin to be configured in the UI. The CustomPlugin.java class calls on one property—app.name.

applicationContext-<name of project>-plugin.xml

Add the following content to the XML file you created:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:tx="http://
www.springframework.org/schema/tx" xmlns:p="http://www.springframework.org/
schema/p" xmlns:util="http://www.springframework.org/schema/util"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://
www.springframework.org/schema/aop" xmlns:context="http://www.springframework.org/
schema/context" xsi:schemaLocation="http://www.springframework.org/schema/
beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.springframework.org/schema/util http://www.springframework.org/
schema/util/spring-util-3.0.xsd http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd http://
www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-
aop-3.0.xsd http://www.springframework.org/schema/tx http://www.springframework.org/
schema/tx/spring-tx-3.0.xsd" default-autowire="byName">
  <import resource="classpath:/META-INF/ephesoft-custom-plugin/applicationContext.xml"/>
</beans>
```

applicationContext.xml

Add the following content to the XML file you created:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd http://
www.springframework.org/schema/util http://www.springframework.org/schema/
util/spring-util-3.0.xsd http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd http://
www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-
aop-3.0.xsd http://www.springframework.org/schema/tx http://www.springframework.org/
schema/tx/spring-tx-3.0.xsd" default-autowire="byName">
  <import resource="classpath:/META-INF/applicationContext-data-access.xml"/>
  <import resource="classpath:/META-INF/applicationContext-batch.xml"/>
  <import resource="classpath:/META-INF/applicationContext-core.xml"/>
  <bean id="CustomPlugin"
class="com.ephesoft.customplugin.ephesoft_custom_plugin.CustomPluginImpl">
    <property name="batchClassPluginConfigService" ref="batchClassPluginConfigService"/>
  </bean>
</beans>
```

```
</bean>
<context:component-scan base-
package="com.ephesoft.customplugin.ephesoft_custom_plugin"/>
</beans>
```


pom.xml

Replace the content of the pom.xml file with the following:

```
<?xml version="1.0"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.ephesoft.customplugin</groupId>
  <artifactId>ephesoft-custom-plugin</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>ephesoft-custom-plugin</name>
  <url>http://maven.apache.org</url>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <dependencies>
    <dependency>
      <groupId>com.ephesoft.dcma</groupId>
      <artifactId>dcma-core</artifactId>
      <version>0.0.1</version>
      <scope>system</scope>
      <systemPath>${project.basedir}/src/main/resources/ephesoft.jar</systemPath>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>3.0.5.RELEASE</version>
    </dependency>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.8.1</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-beans</artifactId>
      <version>3.0.5.RELEASE</version>
    </dependency>
  </dependencies>
</project>
```


Compile the custom plugin project

1. Copy and paste the ephesoft.jar file from <Transact_Folder>\Application\WEB-INF\lib to your project's src/main/resources folder.
2. Right-click the project folder.
3. Select **Run As > Maven build**.
4. Set **Goals** as **clean install**.
5. Click **Apply**.
6. Click **OK**.

 After compiling, you may need to refresh your project to update the folder structure.

Prepare custom plugin for import

1. After compiling your project in Eclipse, locate the .jar file under the target folder.
Example: If the project name is ephesoft-custom, the .jar file is called ephesoft-custom-plugin.0.0.1-SNAPSHOT.jar.
2. Temporarily copy the .jar file to your Desktop.
3. Rename the .jar file to remove <version number>-SNAPSHOT from the file name.
Example: If the .jar file is named ephesoft-custom-plugin.0.0.1-SNAPSHOT.jar, rename it to ephesoft-custom-plugin.jar.
4. Locate <project name>-plugin.xml (such as ephesoft-custom-plugin.xml) under your project.
5. Temporarily copy the .xml file to your Desktop.
6. Compress the .jar and .xml files into a single .zip file.
Example: If you have ephesoft-custom-plugin.jar and ephesoft-custom-plugin.xml, your compressed file should be ephesoft-custom-plugin.zip.


 The .zip file must have the same name as the .jar file.

7. The plugin you created is now ready to import into Transact. Follow these steps:
 - a. In Transact, select **System Configuration > Workflow Management**.
 - b. Drag your created .zip file into the **Import Plugin** section.
 - c. Restart Transact.

The custom plugin you created is now available for use. For more information about importing custom plugins, see the next section.

Import custom plugins

In addition to custom plugins you create, Transact administrators can import custom plugins provided by Kofax and other vendors. This procedure is provided for Transact administrators working in an on-promises environment. For Transact Cloud environments, free custom plugins are available in your instance by default.

 To purchase a premium custom plugin, contact Sales.

Custom plugins are imported as a ZIP file, which contains a plugin JAR file and an XML file with information about the plugin, as shown in [Prepare custom plugin for import](#). Once a custom plugin is uploaded into Transact, the system creates an entry in the database where all default plugins are listed. The plugin .jar is copied to the `customPluginJars` folder in `SharedFolders`, along with the associated .xml file.

To import a custom plugin, perform the following steps. To remove a custom plugin, see [Remove custom plugins](#).

1. In Transact, select **System Configuration > Workflow Management**.
2. In the **Import Plugin** panel, click **Select Files**.
3. From the window that appears, select the .zip file for the plugin and click **Open**.
4. Restart Transact.


The custom plugin is now available for use.

Add plugins to a batch class


To use the plugin, navigate to the associated module, and add the plugin to the Selected Plugins column. For instructions, see the *Ephesoft Transact Administrator's Help*.

Remove custom plugins

To remove custom plugins from Transact, you must remove them from all batch classes and then modify the database.

 Back up the database before removing a custom plugin from your workflow.

1. Identify the affected batch classes using the query below. Any batch classes currently using the plugin need to be redeployed once the plugin is removed from the system.

 In this and the following step, replace `PLUGIN_NAME` with the name of the custom plugin you want to remove.

```
SELECT identifier,
       batch_class_name
FROM   batch_class
WHERE  is_deleted = 0
       AND id IN (SELECT batch_class_id
                  FROM   batch_class_module
                  WHERE  id IN (SELECT batch_class_module_id
                              FROM   batch_class_plugin
                              WHERE  plugin_id IN (SELECT id
                                                    FROM   plugin
                                                    WHERE
                                                          plugin_name LIKE '%PLUGIN_NAME%')));
```

2. Remove all database entries pertaining to the custom plugin by using the following query.

```
DELETE FROM batch_class_plugin_config
WHERE  plugin_config_id IN (SELECT id
                           FROM   plugin_config
                           WHERE  plugin_id IN (SELECT id
                                                FROM   plugin
                                                WHERE
                                                      plugin_name LIKE '%PLUGIN_NAME%'));

DELETE FROM plugin_config_sample_value
WHERE  plugin_config_id IN (SELECT id
                           FROM   plugin_config
                           WHERE  plugin_id IN (SELECT id
```



```
                FROM plugin
                WHERE
                plugin_name LIKE '%PLUGIN_NAME%'));

DELETE FROM batch_class_plugin
WHERE plugin_id IN (SELECT id
                    FROM plugin
                    WHERE plugin_name LIKE '%PLUGIN_NAME%');

DELETE FROM plugin_config
WHERE plugin_id IN (SELECT id
                    FROM plugin
                    WHERE plugin_name LIKE '%PLUGIN_NAME%');

DELETE FROM plugin_dependency
WHERE plugin_id IN (SELECT id
                    FROM plugin
                    WHERE plugin_name LIKE '%PLUGIN_NAME%');

DELETE FROM plugin
WHERE plugin_name LIKE '%PLUGIN_NAME%';
```

3. If AUTOCOMMIT is OFF, issue a commit:

```
COMMIT;
```

- 4. Remove the following files from the customPluginJarsfolder, located at <Transact_Folder>\SharedFolders\customPluginJars:**
 - Custom plugin .jar file (such as plugin-export-multipage.jar)
 - Custom plugin .xml file (such as applicationContext-pluginExportMultiPage.xml)
- 5. Remove the custom plugin folder from the workflow.**

This folder is located at <Transact_Folder>\SharedFolders\workflows\plugins.
- 6. Redeploy the batch classes identified in step 1.**
- 7. Restart Transact.**

Chapter 2

External applications

You can integrate external applications with the Transact Review and Validation modules. External applications are technology-independent and can be written in any language, such as HTML, JavaScript, GWT, JSP, Servlet, or a combination thereof.

Transact interacts with external applications by appending the application's URL with the following two parameters:

- The path of the batch.xml for the current batch using the `batch_xml_path` parameter.
- The document identifier, using the `document_id` parameter.

The batch.xml path is encoded using `java.net.URLEncoder` and UTF-8 encoding. The following is a sample URL for an external application, as fired by Transact:

```
<Ext. App URL>&document_id=<Document Identifier>&batch_xml_path=<Path of batch.xml>&ticket=<Security Token>
```

External applications need to include the following method in their code. They need to invoke this method on the respective button (OK or close) calls they have implemented. External applications signal Transact to perform a specified operation by passing the appropriate operation string in the method argument.

GWT-based applications

```
private native void fireEvent(String operation) /*-{  
    window.top.postMessage(operation, "*");  
}-*/;
```

JavaScript-based applications

```
function fireEvent(var operation) {window.top.postMessage(operation, "*");}
```

The following table describes the actions performed in Transact based on the arguments passed to this method in the external application's code:

Argument Passed by External Application	Result in Transact
Save	The dialog box containing the external application on the Review and Validate screen closes and the changes made in batch.xml are reflected on the screen.
Cancel	The dialog box containing the external application on the Review and Validate screen closes, without refreshing the screen.
Any other string	No change.

Application security

A dynamic token is generated each time an external application is called. This token is sent to the external application by appending the ticket parameter to the URL. Once this token is received, the external application checks the provided URL to determine the authenticity of the token.

Example URL:


```
http://<EphesoftServerIP>:<port>/dcma/authenticate?ticket=<ticket>
```

If the token is not valid, you will receive a 401 error message. A valid token becomes invalid in the following scenarios:

- If the token has already been sent to the Transact server for authentication.
- After an hour has passed since the token was issued.

Add an external application

Perform the following steps to integrate your external application.

 These steps describe how to add the application to the Validate module, but the same can be applied to the Review module if needed.

1. From the **Batch Class Management** screen, select your batch class and click **Open**.
2. Select **Modules > Validate Document > VALIDATE_DOCUMENT**.
3. Set the **External Application Switch** to **ON**.
4. Assign the URL of the application to one of the shortcuts, and add a title. This will be the shortcut used to access the application during validation. The available shortcuts are:
 - Ctrl + 4
 - Ctrl + 7
 - Ctrl + 8
 - Ctrl + 9

Access the application

From the Review or Validate screens, use your configured shortcut to call the external application. You can also call the application manually by clicking **More > External Application** and selecting your application from the list. Your application opens in a separate window.

Transact AI Table Rule Builder

The Ephesoft Transact AI Table Rule Builder enables operators to create extraction rules for invoice table line items during the Validation stage in Transact. This feature also provides

improved extraction accuracy by excluding rows that are not needed. With AI Table Rule Builder, users can configure and save extraction rules that are unique to vendor-specific invoices. The Transact AI Table Rule Builder is provided as an external application that is configured in the VALIDATE_DOCUMENT plugin and integrated with the Validation module.

i The Transact AI Table Rule Builder is currently limited to creating rules for single-page basic invoices. It was designed for invoice document types, and has not been tested with other document types.

Prerequisites

To use the Transact AI Table Rule Builder, the following prerequisites must be in place:

- Transact must be installed.
- Install and enable the Transact AI Table Rule Builder. Follow the steps in the InstallationNotes.txt packaged with the Transact AI Table Rule Builder .zip file.
- Ensure the TABLE_EXTRACTION plugin is added to the Extraction module and turned on.
- Add the table and table columns to your document type. Transact can only create extraction rules for existing tables. For steps, refer to "Add table columns" in the *Ephesoft Transact Administrator's Help*.

⚠ Each Column Name must exactly match the column header for the table; otherwise, the table is not extracted and the rule is not created.

Installation of Transact AI Table Rule Builder

1. Download the current version of the Transact AI Table Rule Builder plugin.
2. Stop the Transact server.
3. Extract the .zip file to a temporary location.
4. Install the Transact AI Table Rule Builder plugin for your operating system as described in the table.

Operating System	Procedure
Windows	<ol style="list-style-type: none"> a. Navigate to the folder where you extracted the plugin files. b. Open a command prompt and run the following command: <pre>install_auto_table_rule_builder.bat</pre>
Linux	<ol style="list-style-type: none"> a. Navigate to the directory where you extracted the plugin files. b. Open the terminal and run the following command: <pre>./install_auto_table_rule_builder.sh</pre>

Operating System	Procedure
	<p>i If you are unable to run the command because of a permission issue, run the following:</p> <pre data-bbox="951 478 1466 533">chmod +x install_auto_table_rule_builder.sh</pre>

5. Start the server.

i For Transact Cloud users, the Technical Operations Team installs the Transact AI Table Rule Builder for you. However, you will need to configure the VALIDATE_DOCUMENT plugin. For configuration instructions, see the VALIDATE_DOCUMENT plugin instructions in the *Ephesoft Transact Administrator's Help*.

Configure the VALIDATE_DOCUMENT plugin

You must configure the VALIDATE_DOCUMENT plugin to use the Transact AI Table Rule Builder. Once configured, the Transact AI Table Rule Builder appears in the Validate screen as a selectable link from the **More External Application** menu.

1. From the **Batch Class Management** page, select your batch class and click **Open**.
2. Expand the **Validate Document** module and select the **VALIDATE_DOCUMENT** plugin.
3. In the Plugin Configuration screen, set the **External Application Switch** to **ON**.
4. Type `http://<Server_Name>:8080/dcma/autoTable` or an HTTPS URL in one of the following fields:
 - **URL1(Ctrl+4)**
 - **URL2(Ctrl+7)**
 - **URL3(Ctrl+8)**
 - **URL4(Ctrl+9)**
5. In the corresponding **URL Title** field, type a name for the **External Application** link.
6. Click **Deploy**.

Launch the application

Operators can launch the application from the Validation page for a batch instance by clicking **More > External Application** and then selecting the link assigned to the Transact AI Table Rule Builder.

Operators can use shortcuts to launch the AI Table Rule Builder. The shortcut is dependent on which External Application URL is associated with the AI Table Rule Builder. For example, if it is registered as the first External Application URL for the batch class, then the shortcut **CTRL+4** opens it. Other shortcuts are as follows:

External Application URL	Shortcut
URL1	Ctrl+4

External Application URL	Shortcut
URL2	Ctrl+7
URL3	Ctrl+8
URL4	Ctrl+9

Table Extraction Rule Builder modes

Using one of four modes (Automatic, Standard, Advanced or Manual), operators can select the configuration mode that best suits their skill level. The guided in-app configuration screen helps users identify line item information to extract. Transact then creates the extraction rules and uses them to extract data from any subsequent invoices submitted by the same vendor.

The Table Extraction Rule builder includes multiple modes depending on your needs.

Extraction Mode	Description
Automated mode	This is the AI Table Rule Builder default mode. If you need to edit the extraction results or rule, you can select one of the other modes to make changes.
Standard mode	Recommended edit mode for most use cases. This wizard walks you through each step in configuring an extraction rule.
Advanced mode	Alternative edit mode for operators, administrators, or developers familiar with regular expressions.
Manual mode	Use this mode to extract table data without creating new table extraction rules.

Automated mode

Automated Mode is the default configuration mode for Transact AI Table Rule Builder. Operators define the location of the table data on the document using an overlay, select a table name from a menu, and click **Next**. Then, Transact uses an AI engine to extract and analyze the data and generate extraction rules. If needed, operators can manually adjust column names or select an alternative mode (Standard, Advanced, or Manual) to refine the new extraction rule. Once configured, Transact populates the table Validation screen using the new table extraction rule. Operators can then finish validating the extracted data.

To create an extraction rule using the wizard:

- From the field, select the table for which you want to build the rule.
If no tables are listed, or if your desired table is missing, make sure you completed the requirements listed in [Prerequisites](#).
- Use the overlay to draw a box around your table on the preview image. This should include the header row, and span the full height and width of the table.
- Click **Next**.
- Review the extracted results.
 - If you are satisfied with the results, click **Next** to review the extraction rule, then click **Save**. Your extraction rule is created and the wizard is closed.

- If the columns contain the correct data, but are mapped incorrectly, select the correct header name using the label button and proceed as normal.

Note these important items:

- When adding a new table and table columns for a document type, the **Column Name** must exactly match the column header, or the table is not extracted, and the rule is not created. Alternatively, the dictionary synonyms should be defined to properly match the column header. See [Automated Mode Dictionary](#) for more information on configuring the dictionary synonyms.
- When using Automated Mode, column data may merge together if data in columns are closely aligned. See the *Ephesoft Transact Release Notes* for more information.
- If the columns do not contain the correct data, or are missing data, click **Edit** and select your edit mode.

Select one of the following edit modes based on your use case:

- Standard mode (recommended)
- Advanced mode
- Manual mode

Automated Mode Dictionary

Batch class administrators can use the Automated Mode Dictionary to define a set of table header synonyms. The Transact AI Table Rule Builder uses synonym-based detection if it is unable to detect headers by matching column names using the Transact table header detection algorithm.

To configure the Automated Mode Dictionary:

1. From the **Batch Class Management** page, select your batch class and click **Open**.
2. Go to **Document Types** > <Document Type name> > **Tables** > <Table Name> > **Table Columns**.
3. In the **Column Synonym** column, create regex patterns for each synonym.
4. Click **Deploy** once you have created the synonyms.

Standard mode

Standard mode is the recommended edit mode for most use cases. Standard Mode uses a guided wizard to dynamically build table extraction rules. Once a rule is created, operators can review the rule and tune it for improved extraction results. Standard mode also includes the option to select any rows in the table that should be excluded from the extraction rule.

This wizard walks you through creating an extraction rule for your table step by step. To access this wizard, click **Edit** when available and select **Standard Mode (recommended)**.

The wizard separates creating an extraction rule into the following steps:

Step in Wizard	Action
Step 1: Header Row	Select whether a header row exists for this table or not. If yes, draw the provided overlay over the full height and width of the header row. The header row is the preferred option when creating an extraction rule.

Step in Wizard	Action
Step 2: Preceding Text	Select whether the table has preceding text (text immediately before the table). If yes, draw the overlay over a single line of preceding text. Note: The system needs some header row or preceding text to create an extraction rule. If both the header row and preceding text are selected, then Transact will default to using the header row text to create the extraction rule. For help, see Preceding and trailing text .
Step 3: Trailing Text	Select whether the table has trailing text (text immediately after the table rows). If yes, draw the provided overlay over a single line of trailing text. For help, see Preceding and trailing text .
Steps 4 and up: Columns (<i>varies depending on number of columns</i>)	Select whether the table has the mentioned column. If yes, draw the overlay over the full height and width of the column, not including the header. This step will repeat based on the number of columns configured for the selected table.
Final step: Ignore Rows	Select whether any extracted rows should be ignored using the sample extracted table. For example, on an invoice you may want to ignore the subtotal rows. Note: If you reach this step and no data is populated in the sample table, Transact is not able to create an extraction rule. Try the wizard again, or use Automated mode Otherwise, you can perform a one-time extraction using Manual mode .

Preceding and trailing text

Identifying preceding and trailing text helps Transact determine where a table begins and ends. Use the following guidelines to select the best preceding and trailing text:

- This text should be immediately before or after your table. If anything is between your text and the table it is treated as part of the table, and may result in incorrect extraction results.
- This text should be static, which means it does not change from document to document.
- This text should only span a single line.

The following are examples of good and poor images for extracting trailing text.

Good images

The following example shows a sample invoice with the line Sub Total selected as the trailing text. This works well for this invoice because:

- There is nothing between the table and the highlighted text.
- The text is an invoice field name, which means it does not change.
- The text only spans a single line.

INVOICE

ACME Company
 Belverly Hills Blvd
 Irvine, CA 90210
 Tel: 949-331-7500

Invoice No: 5432000
 Invoice Date: 04/06/08
 PO Number: 2005012345

Part No	Quantity	Unit Price	Description	Discount	Total
998100000156	50	\$0.16	FS B&W Card Stock	0.00%	\$8.00
9981000001990	9	\$1.09	Cutting-Per Reim	0.00%	\$9.81
				Sub Total	\$17.81
				Tax 7.75%	\$1.38
				Total	\$19.19



Thank you for your business.
 MAIN Corporate Office

Page 1 of 1

Poor images

The following example shows the same sample invoice with the line "Thank you for your business" selected as the trailing text. This is a poor selection because there are multiple lines between this line and the table, including a QR code.

INVOICE

ACME Company
 Belverly Hills Blvd
 Irvine, CA 90210
 Tel: 949-331-7500

Invoice No: 5432000
 Invoice Date: 04/06/08
 PO Number: 2005012345

Part No	Quantity	Unit Price	Description	Discount	Total
9981000000156	50	\$0.16	FS B&W Card Stock	0.00%	\$8.00
9981000001990	9	\$1.09	Cutting-Per Reim	0.00%	\$9.81



Sub Total: \$17.81
 Tax 7.75% \$1.38
 Total \$19.19

Thank you for your business
 MAIN Corporate Office

Page 1 of 1


Advanced mode

Advanced mode provides operators with additional flexibility to create extraction rules using column overlays and custom regex patterns. Operators can also test regex patterns within the rule builder or exclude rows from the rule for additional fine tuning. This mode is ideal for users with advanced knowledge of regex.

To access this mode, click **Edit** when available and select **Advanced Mode**.

To create an advanced rule using regular expressions:

1. Create the following regular expressions (regex):
 - **Start Pattern Regex:** Defines the starting point of the table. This must be unique across all extraction rules in a document type. A good choice for a start pattern is often part of the table column headers. For example, "Code Description Price."
 - **End Pattern Regex:** Defines the end point of the table. See [Preceding and trailing text](#) for more information.
 - **Column Regex:** Defines the pattern for the column data. You will need to define a regular expression for each available column.


 Each column includes selectable **Anchor** and **Required** options.

Checkbox	Description
Anchor	This option enables you to indicate the start of a new row if a value from this column is extracted. This is useful when table rows span more than a single line and enables text to wrap. For best results, select a single value from the end of the wrapped line that will always be present. You can only select one Anchor checkbox at a time.
Required	Selecting the required checkbox will make the column mandatory for operator validation. The extraction algorithm will always expect this column to be present in the document.

- **Row Exclusion Regex:** Define a pattern for table rows that should be ignored, such as the subtotal row in an invoice. If a row is extracted that fully or partially matches the regex, that row will be removed from the table results.
2. Resize the provided overlays over the full height and width of each column. These overlays are color-coded and labeled to match each column.
 3. Click **Test Extraction** and review the extracted results. You may need to test multiple times as you fine-tune your rule.
 4. When you are satisfied, click **Save**. The extraction rule will be created and you will return to the **Validation** screen.

Manual mode

Manual mode enables users to extract table data without creating new table extraction rules. It is ideal to use when operators need to quickly extract and validate data but no rule exists, documents have poor OCR quality, or contain a complex table layout. Although Manual mode will extract and save data to the batch instance, it does not create and save extraction rules to the batch class.

 Using this mode will *not* create an extraction rule for the table.

Refer to the following section for help performing manual edits. When you are satisfied with your changes, click **Update Batch** to return to the Validation screen.

Performing manual edits


The following section provides a summary of the table editing tools and how to use them.

Edit cells

Click any non-header cell to enter edit mode and begin typing.

Map incorrect headers

Fix any incorrect headers by clicking the label icon and selecting a header from the field.

 A header can only be mapped to one column at a time. Selecting a header that is already mapped to another column will swap the mapping with the existing column.

You can also insert, merge, or delete rows.

Insert a row

Click the three-dot icon to the right of a row and select either **Insert row above** or **Insert row below**.

Merge rows

Click the three-dot icon to the right of a row and select either **Merge row above** or **Merge row below**.

Delete a row

Click the ellipsis icon to the right of the row you want to delete and select **Delete row**.

You can also clear an entire column.

Clear column

Click the three-dot icon to the right of a row and select **Clear column**.

Limitations

The following types of tables are not good candidates for the Transact AI Table Rule Builder, and may receive inaccurate results:

- Tables with overlapping columns.
- Tables with closely packed column data.
- Tables within the cell of another table (such as nested tables).
- Tables with hidden columns.
- Tables that span more than one page.

Chapter 3

Regular expressions

Transact uses regular expression patterns in the Extraction module to search, edit, or manipulate text and data. Regular expressions describe a set of strings based on common characteristics shared by each string in the set.

The following table lists basic regular expression constructs.

Construct	Description
[abc]	a, b, or c (simple class)
[^abc]	Any character except a, b, or c (negation)
[a-zA-Z]	a through z, or A through Z, inclusive (range)
[a-d[m-p]]	a through d, or m through p: [a-dm-p] (union)
[a-z&&[def]]	d, e, or f (intersection)
[a-z&&[^bc]]	a through z, except for b and c: [a-d-z] (subtraction)
[a-z&&[^m-p]]	a through z, and not m through p: [a-lq-z] (subtraction)

Predefined character classes of regular expressions

Commonly used regular expressions are organized into a set of predefined character classes. The following table lists the constructs assigned to the predefined character classes.

Construct	Description
.	Any character (may or may not match line terminators)
\d	A digit: [0-9]
\D	A non-digit: [^0-9]
\s	A whitespace character: [\t\n\x0B\f\r]
\S	A non-whitespace character: [^\s]
\w	A word character: [a-zA-Z_0-9]
\W	A non-word character: [^\w]

Quantifiers

Quantifiers enable you to specify the number of occurrences to match against.

Pattern	Meaning
X?	X, once or not at all
X*	X, zero or more times
X+	X, one or more times
X{n}	X, exactly n times
X{n,}	X, at least n times
X{n,m}	X, at least n but not more than m times

Capturing groups

Capturing groups treat multiple characters as a single unit. They are created by placing the characters to be grouped inside a set of parentheses. For example, the regular expression (dog) creates a single group containing the letters d, o, and g. The portion of the input string that matches the capturing group will be saved in memory for later recall with backreferences.

Capturing groups are numbered by counting their opening parentheses from left to right. In the expression ((A)(B(C))), for example, there are four such groups:

- ((A)(B(C)))
- (A)
- (B(C))
- (C)

Backreferences

The section of the input string matching the capturing group(s) is saved in memory for later recall with backreferences. A backreference is specified in the regular expression as a backslash (\) followed by a digit indicating the number of the group to be recalled. For example, the expression (\d\d) defines one capturing group matching two digits in a row, which can be recalled later in the expression via the backreference \1.

For example, to match any 2 digits, followed by the exact same two digits, use (\d\d)\1 as the regular expression:

Regular expression	(\d\d)\1
Input string	1212
Result	Found the text "1212" starting at index 0 and ending at index 4.

Capturing groups and character classes with quantifiers

Examples:

- (abc)+ (the group "abc", one or more times).
- [abc]+ (a or b or c, one or more times)

Boundary matchers

With boundary matchers the location of the match can be found within a particular input string (for example, the beginning or end of a line), on a word boundary, or at the end of the previous match. The following table lists and explains all the boundary matchers.

Boundary Construct	Description
^	The beginning of a line
\$	End of a line
\b	A word boundary
\B	Non-word boundary
\A	Beginning of the input
\G	The end of a previous match
\Z	The end of the input for the final terminator, if any
\z	The end of the input

Examples:

Regular expression	^dcma\w*
Input string	dcma ephesoft
Match Found	true

Grouping constructs

Use grouping constructs to capture groups of sub-expressions and to increase the efficiency of regular expressions with non-capturing lookahead and lookbehind modifiers. The following table describes the Regular Expression Grouping Constructs.

Grouping Construct	Description
(?i)	Turn on case insensitivity for the remainder of the regular expression. (Older regex flavors may turn it on for the entire regex.) For example, te(?i)st matches teST but not TEST.
(?:)	Non-capturing group.

Grouping Construct	Description
(?=)	Zero-width positive lookahead assertion. Continues match only if the sub-expression matches at this position on the right. For example, <code>\w+(?=\d)</code> matches a word followed by a digit, without matching the digit. This construct does not backtrack.
(?!)	Zero-width negative lookahead assertion. Continues match only if the sub-expression does not match at this position on the right. For example, <code>\b(?!\un)\w+\b</code> matches words that do not begin with un.
(?<=)	Zero-width positive lookbehind assertion. Continues match only if the sub-expression matches at this position on the left. For example, <code>(?<=19)99</code> matches instances of 99 that follow 19. This construct does not backtrack.
(?<!)	Zero-width negative lookbehind assertion. Continues match only if the sub-expression does not match at the position on the left.

Sample regular expressions:

- **Regular expression for email address:** `[_A-Za-z0-9-]+(\.[_A-Za-z0-9-]+)*@[A-Za-z0-9]+(\.[A-Za-z0-9]+)*(\.[A-Za-z]{2,})`
- **Regular expression for date:** `(0[1-9]|1[012])[- /.](0[1-9]|12)[0-9]|3[01])[- /.]\d\d([0-9]{2})?`
Matches the date in the following formats: mm/dd/yyyy or mm.dd.yyyy or mm-dd-yyyy or mm/dd/yy or or mm.dd.yy or mm-dd-yy
- **Regular expression for time in 12-Hour Format:** `(1[012] | [1-9]):[0-5][0-9](\s)?(am | pm)`
Matches the time in the following format: 12:45am or 1:34pm or 7:56AM or 2:57PM or 1:45 PM or 2:34 AM
- **Regular expression for the price:** `\d+[,]{0,1}\d+[\.]?d{1,2}`
Matches the prices in the following formats: 123.89 or 12,889.90 It will not match a single digit or two digit prices.
- **Word boundary match example:** To match whole word only, "`\b`" is used in the regular expressions. For example, to match a word "dcma" but only if it is whole word, for the input data: "dcma dcmaEphesoftData," use the regular expression listed below.
- **Lookahead and lookbehind example:** To match something not followed or preceded by something else, use lookahead and lookbehind assertions. Matching "date" not preceded by "due."

Input data	due date is 22/11/2012 and the end date is 11/11/1999
Regular expressions	<code>(?<! due\s)date</code> <code>\bdcma\b</code>
Matching	"date" not followed by "due"

Input data	Payment date due is 22/11/2012 and actual date is 11/11/1999
------------	--

Regular expression	date(?:\sdue)
Matching	In both the cases there will be only one match of "date" string.

- **Pattern matching two words near each other:** This pattern consists of three parts:
 - The first word
 - A certain number of unspecified words
 - The second word

An unspecified word can be matched with the shorthand character class '\w+'. The spaces and other characters between the words can be matched with '\W+' (uppercase W this time). Finding any pair of two words such as "payment" and "bank" in the data:

Regex pattern	payment\W+(?:\w+\W+){1,6}?bank
Matching	Pair of words (payment, bank) separated by at least one word and at most six words between them.

Regex patterns in Transact

In table extraction, regular expressions support multi-word capture. In KV extraction, regular expressions use word-based extraction.

Usage of 'pattern' field in 'document Index Field Details'

In the document index field details, the administrator can enter some comma-separated values in the "pattern" field. The last value in the pattern field is a regular expression used to match the data and the previous values are used as key values. There may be multiple matches for the regex pattern but you want only those matches preceded by some specific values (key values specified in the pattern list).

Example:

Regex pattern	Invoice; Date; [0-9]{2}/ [0-9] {2}/[0-9]{2,4}
Matching	Only dates preceded by the strings "Invoice" and "date".

Usage of multi-word in key pattern for k-v extraction

The multi-word capturing in KV extraction is present only in the key extraction and not in value extraction. For example, to capture the value "22/09/2011" for the input data: Invoice date 22/09/2011 The following key and value patterns can be used.

Key pattern	Invoice date
Value pattern	(0[1-9] 1[012]) [- ./](0[1-9] [12][0-9] 3[01])[./]\d\d ([0-9]{2})?

How not to capture certain values for key pattern and value pattern

To match something not followed or preceded by something else, use lookahead and lookbehind assertions. For example:

Input data	due date is 22/11/2012 and end date is 11/11/1999
Regular expression	(?<!due\s)date
Matching	"date" not preceded by "due"

Input data	Payment date due is 22/11/2012 and the actual date is 11/11/1999
Regular expression	date (?!\s)due
Matching	"date" not followed by "due"

In both, the cases there will be only one match of "date" string.

Usage of multi-word capture in Table Extraction, which is different than Value Pattern in K-V Extraction

Multi-word capture in Table Extraction is different from the value pattern in Key-Value Extraction. For example, consider the following image data:

Date	Product	Quantity	Price
11/22/2012	iPod Touch	5	25000.00
05/22/2012	Laptop	2	30000.50

In this table, the multi-word data "iPod touch". can be captured using the regular expression: [A-Za-z\s].

But in K-V Extraction, multi-word data capturing is not supported for "value" pattern.

Chapter 4

Scripting resources

This chapter covers resources related to scripting in Transact.

Scripting functionality

Scripting functionality in Transact is supported where a custom requirement can be handled. Transact includes power to perform custom functionality during different stages of batch processing. This functionality is supported through the scripts present inside the batch class folder in the `Scripts` folder. These scripts are Java scripts which access `batch.xml` through two techniques: DOM Parser and JDOM Parser. All the scripts present should be developed using one of the earlier mentioned parser only.

Batch Instance Group feature

This feature is used to provide the roles on the batch instance. If a user role has access to a batch instance using batch instance group feature than that batch will display on the BatchList screen. User will able to review/validate the same batch instance.

The database table `batch_instance_groups` is used to store the batch instance identifier and all the mapped user roles with the batch instances. The following database structure is used.

Method for assigning roles in batch instance group table

```
public void assignedBatchInstanceGroup(String batchInstanceIdentifier, String userRole)
    throws DCMAException
```

Configurable parameters

Example file: [ScriptDocumentAssembler_BatchInstanceGroupFeature.java](#)

Force Review feature

The Force Review feature is implemented as follows:

- The ForceReview tag is added to `batch.xml` for any document-level fields. This tag has the following characteristics:
 - It is only used when a document is invalid. When it does, the ForceReview tag is set to true.
 - It is not a mandatory tag. If the ForceReview tag does not exist for that document class, or the tag is not set to true, the document is processed as normal.

- When this batch opens for validation, and there is an invalid document that sets the ForceReview tag to true, the following happens:
 - Before the user saves the document (either by pressing Ctrl+S or Ctrl+Q), the invalid field appears red, regardless of Regex validation.
 - Once the user presses Ctrl+S or Ctrl+Q on a field, the ForceReview tag is set to false for the field. Validation can continue as usual (such as by using Regex validation).
- The script must handle the setting of the ForceReview tag in sync with the document validity setting.
- The ForceReview tag of a field does not make a document invalid. It will only prevent a document-level field from getting validated in a document in an invalid state if set to true.
- This ForceReview tag does not exist by default (before validation).

Provided is a sample script for the generation of this ForceReview tag(or modifying it if it is already present) with a value "true" for every alternate(note the i+=2 in the script) document level field encountered in a batch.xml.

Example file: [Force_Review_Feature_SampleScript.java](#)

ScriptAddNewTable.java

The Add New Table script is useful for being able to generate a predefined table on the fly in the Validation view.

Example Scenario: If you are processing invoice documents and have defined table extraction during the normal means, it is still possible that if the document is of poor image quality the table you have defined for extraction may not be detected, or may be incomplete. In this case, you can use the Add New Table script to automatically generate a table for your invoice document type with the click of a button. You could create a new empty table of your required structure, or even a populated one, using other extracted values from the documents fields to populate the table values from a database table.

Example file: [ScriptAddNewTable.java](#)

ScriptAutomaticValidation.java

The Automatic Validation script is executed in the Transact batch class workflow prior to reaching the Validation stage. This is the opportune time to implement any custom logic you may require by running verification against the values that have been extracted for the fields defined.

Example Scenario: You may wish to retrieve the extracted value from one field, perhaps a Social Security Number, and use it to perform a look-up operation to an external database, retrieving an associated name. You could then in turn use the results of these SQL query to populate the values of other document level fields defined in Transact.

Example location: <Transact_install_directory>\SharedFoldersBC1scripts

Example file: [ScriptAutomaticValidation.java](#)

ScriptDocumentAssembler.java

The Document Assembler script is executed in the Transact Batch Class work flow following the Page Processing Module, and Prior to reaching the Document Review stage. At this point the batch .xml has undergone Transact's classification routines and the batch's contents are structured into documents, with confidence scores assigned. However, perhaps you have a unique situation where the default Transact classification behavior alone will not organize your batch's documents as you require. This script is the ideal location to implement any logic to customize the classification organization of your batch. Although Document Level Fields are not yet present in the batch .xml at this point, page level fields are making Script

Example Scenario: You may be using cover sheets to aid in the classification accuracy for your documents. However by the time the work flow reaches the ScriptDocumentAssembler Transact's classification routines have already worked their magic and you may no longer have any use for the cover sheets. You can use this script to define some logic to remove the first page of every document, which in this case would be the cover sheets.

We provided as an example resource, a script that does exactly this:
[ScriptDocumentAssembler_remove-1st-page.java](#)

ScriptExport.java

The Export script is executed in the Transact Batch Class work flow during the processing of the Export Module. At this point all of Transact's separation, classification, extraction, and validation have been performed and the batch is in the process of leaving the Transact system. Consequently, ScriptExport.java is the ideal place to implement logic facilitating custom export requirements.

Example Scenario 1: As a batch is leaving Transact, you may intend to import its .xml and image files into a document repository system. Your targeted system may have its own internal XML schema for describing its contents, and to make the transition a seamless one you want to consider applying an XSLT translation to the batch.xml so that it can be automatically imported and recognized by its destination system without manual effort.

Example Scenario 2: Perhaps you have a specific document management system such as FileBound in mind. FileBound uses Divider and Separator values to index its contents. Transact provides a FileBound Plugin in the Export Module. However, in addition to this you can implement logic in the ScriptExport.java file to populate the batch documents' divider and separator fields based on a captured extraction value, barcode, or other means. Through Transact scripting, you can use an extracted bar code value as a look up key in a SQL query to an external database table to retrieve the appropriate divider and separator values for a given document type.

Example location: <Transact_install_directory>\SharedFoldersBC1scripts

Example file: [ScriptExport.java](#)

ScriptExtraction.java

The Extraction script, although similar in sequence position to the ScriptAutomaticValidation.java, is useful for the purpose of separating extraction from validation.

Example Scenario: Assume you have some OMR field extraction defined to extract check boxes denoting credit card type. For example there could be three boxes total, signifying a choice of Visa, American Express, or MasterCard. If filled out correctly the form should only have a check mark in one of the boxes. RecoStar OMR extraction will represent these three check boxes as a three character string of binary values, with the box containing a mark being represented by a 1 value (the others, 0). Your Extraction script function could analyze the binary values to determine which of the three credit card vendors is being represented and set a document level field value to a string value of visa, amex, or mastercard.

Example location: <Transact_install_directory>\SharedFoldersBC1scripts

ScriptFieldValueChange.java

The Field Value Change script can be toggled through the Validation Module, and when enabled will fire when the value of a field is changed in the user Validation View. This can be extremely useful if you want to have the changes you make to one field automatically produce a change in value of one or more other fields.

Example Scenario 1: Consider the case where you have a field that is of the drop down list type. This field contains a list of various departments within an organization (eg: HR, IT, etc). In addition to this field you have another drop down field that serves as a list of users. You can use the Field Change script to implement logic such that change the department value for field #1 will automatically update the drop down list of field #2 with users that correspond to that newly selected department.

Example Scenario #2: Perhaps you have had a document arrive in Validation View with a loan number field that has been extracted, and some corresponding fields that contain information related to the loan number. However, consider if the loan number is incorrect, and your user manually needs to change it. Instead of having to also manually look up and change all of the corresponding fields, the Field Change Value script could make it so that changing the Loan Number field value automatically performs a look up operation to a database and populates the additional fields with the information on file for that loan number.

Example location: <Transact_install_directory>\SharedFoldersBC1scripts

Example file: [ScriptFieldValueChange.java](#)

FunctionKey.java

The Function Key script allows for multiple methods to be defined within the FunctionKey.java and have these individual methods bound to keys on the users keyboard to act as keyboard shortcuts for executing pieces of custom functionality.

Example Scenario: Consider that you may have a default set of values for a given document type that you want the option to populate its fields with in Validation View at the push of a button. In FunctionKey.java you would implement a method to set the fields of that document type to a set of default values. In the Validation Module you can define the Function Key that you want to associate with this method. You must specify both the name of the function as well as the key that you wish you bind it to. Having done so you should now see the Function Key that you specified as an icon in the Validation View. The custom function key method can now be invoked by clicking this button or striking the specified key itself.

Example location: <Transact_install_directory>\SharedFoldersBC1scripts

Example file: [ScriptFunctionKey.java](#)

ScriptPageProcessing.java

The Page Processing script is useful for conducting operations on the XML representation of a batch's pages before they have undergone classification and encapsulation into documents.

Example Scenario: To reduce batch processing time it is possible to use ScriptPageProcessing to remove pages that are garbage and does not contain valid OCR data. You could implement a function that would iterate through the collection of pages, and from each page element retrieve the corresponding OCR file (.html) and image file name. If the OCR data for the page is nonexistent, and the original image size is below a certain threshold you can deem the page to be junk and remove it from the batch.xml. By removing these useless pages your batch will be able to traverse the rest of the batch class workflow more efficiently and with an improved processing time.

Example location: <Transact_install_directory>\SharedFoldersBC1scripts

ScriptValidation.java

The Validation script differs from the Automatic Validation script in that it is run during the user Validation User Interface experience, where as Automatic Validation executes and performs custom validation steps immediately before the batch is presented to the user in the Validation UI. The benefit is this is that as the user is making changes to the document field values in the Validation UI, the Validation Script can be triggered to run after each change to ensure that the changes are put through custom validation processing requirements.

Example Scenario: You may have some a document level field, AccountNum, where the value being extracted should correspond to an existing account number in an external database. Perhaps due to poor image quality, the OCR'd value for the account number is either incomplete or incorrect and so the document appears in Validation with its field highlighted in red. The user is able to see the corresponding image and keys in what they believe to be the account number appearing on the image, but you want to ensure that they haven't made a mistake and what they are inputting is indeed a valid account number. In this case it is possible to implement a function that takes the value of the AccountNum field and performs a database look up to verify that the value does in fact exist. This check can run each time after the user has modified the AccountNum document field value and attempts to save their changes.

Example location: <Transact_install_directory>\SharedFoldersBC1scripts

Application Level script

The Application Level script is global to the Transact environment and is fully configurable to add additional logic. Transact will execute the Application Level script at the system or service level. Some common customizations to the script include:

- Running stored procedures in the database.
- Restarting batch instances upon error.

- Executing custom imports into a batch instance.
- Sending email notifications to users.

The script is executed by a cron schedule, which sets how frequently Transact executes a task. To configure an Application Level script, perform the following steps:

1. Write the script.
2. Determine how often Transact executes the script.
3. Add the script to the `dcma-scripting-plugin.properties` file, located in `<Transact Installation Folder>:\Ephesoft\Application\WEB-INF\classes\META-INF\dcma-scripting-plugin\dcma-scripting-plugin-properties`.

This feature provides users with an option to execute a script repeatedly in specific time intervals to automate application jobs and execute a script without initiating a batch.

Create the Application Level script file

The script file is a simple java class implementing an `ApplicationScripts`

1. To execute a script at the application level, import the script project and define what the java class implements and executes.

The following is a sample of the script file structure:

```
import com.ephesoft.dcma.script.ApplicationScripts;

public class MyApplicationLevelScript implements ApplicationScripts {

    public Object execute(String shareFolderPath) {

        /*****task to be done *****/

    }

}
```

2. Save the file with a `.java` extension.

i A default script file will be available in the `application-scriptfolder` in located in `Transact SharedFolders` for reference.

Configuration

1. Specify the following properties in the `META-INF/dcma-scripting-plugin/dcma-scripting-plugin.properties` file:
 - script file name
 - folder name
 - cron expression
2. Save and close the file.

i Application Level scripts are enabled by default. The script on/off switch (`script.scriptSwitch`) available in the properties file does not control the execution of the Application Level script. It only enables or disables execution of batch class level scripts.

Execution

1. Navigate to Transact `SharedFolders` and create a folder with the same name as configured in the property file.
2. Create a script file with the same name as specified in property file.
Save the file with `.java` extension.
3. Create a folder in Transact `SharedFolders` with the same name as specified in property file.
4. Copy the script file to the new folder.
5. You have configured and set up the Application Level script on your Transact server.
The script will begin executing at your specified time interval.

Dependency

For this feature to function in a multi-server environment with the restore mechanism, you will need the heartbeat module to be running on all the servers.

Troubleshooting

Following error messages may appear in the log files due to issues with configuration and execution:


Error Message	Possible Cause
Script not found	Either the script file or folder doesn't exist as specified in the property file. Verify the script folder and folder name present in the Ephesoft SharedFolders.
Script errored out	The script file was not correct. Verify the script file syntax. It must implement an <code>ApplicationScripts</code>
Failed to compile	A java compilation error occurred in the script file. Verify the java code in a java editor.

Batch Instance Group

Users can assign a group to every batch instance using scripts by making entry in the `batch_instance_groups` table in database. The assigned group will only have access to that batch instance.

In case there is no group assigned to the batch instance in the `batch_instance_groups` table, users belonging to the groups in the corresponding batch class will have access to that batch instance.

Users belonging to the group mapped in `batch_instance_groups` will be given preference over the users belong to the group specified in batch class. For example, if a particular batch instance, say BI1, is mapped to group A in `batch_instance_groups` table and corresponding batch class, say BC1 is mapped to group B, in that case group A will be given preference and only users belonging to that group will be able to access the batch instance.

 The only way to utilize this functionality is by script.

Batch.xml and XSD schema

You can modify batch.xml and the XSD schema to add functionality and customize solutions beyond a regular Transact installation.


The batch.xml file schema and matching XSD contain metadata and multi-level information for every batch processed in the Transact workflow. The batch.xml file contains metadata for each batch instance at the batch level, document level and page levels.

The batch.xml file and XSD support the following field levels. This hierarchy of fields applies to the batch.xml schema for each batch instance that has begun the workflow process:

- **Batch-level fields** The fields on this level apply to the entire batch instance as a whole.
- **Document fields:** The fields on this level apply to all documents in the batch instance.
- **Document-level Fields:** The fields on this level apply to individual documents within the batch instance.
- **Page Fields:** The fields on this level apply to all pages within the batch instance.
- **Page-Level Fields:** The fields on this level apply to each individual page within the batch instance.
- **Email Metadata in the Batch.xml Schema:** Email heading metadata is available on multiple levels of the batch.xml schema for any batch instance that uses email import.

Batch-level fields

Refer to the following table for the batch-level fields in the Transact batch.xml schema and XSD.

 For information on email metadata, refer to the [Email metadata in the batch.xml schema](#) section of this document.

Batch-Level Field Name	Description	Module	Plugin
BatchInstanceIdentifier	This value is the Identifier column in the batch_instance table. Each batch in Transact has a unique batch identifier.	Folder Import	IMPORT_BATCH_FOLDER_PLUGIN
BatchClassIdentifier	This value is the Identifier column in the batch_class table. Each batch in Transact is run under a batch class that is a single unit for all configurations and workflow definitions.	Folder Import	IMPORT_BATCH_FOLDER_PLUGIN

Batch-Level Field Name	Description	Module	Plugin
BatchClassName	This value is the batch_class_name column in the batch_class table. Each batch in Transact is run under a batch class that is a single unit for all configurations and workflow definitions. A foreign key relation is established between the ID column of the batch_class table and the batch_class_id column in the batch_instance table.	Folder Import	IMPORT_BATCH_FOLDER_PLUGIN
Signature	This value is added only when batch encryption is enabled. This is used to ensure that the batch.xml file can only be read and updated by Ephesoft Transact.	Folder Import	IMPORT_BATCH_FOLDER_PLUGIN
BatchClassDescription	This is the value of the batch_class_description column in the batch_class table. Each batch in Transact is run under a batch class that is a single unit for all configurations and workflow definitions. A foreign key relation is established between the ID column of the batch_class table and the batch_class_id column in the batch_instance table.	Folder Import	IMPORT_BATCH_FOLDER_PLUGIN

Batch-Level Field Name	Description	Module	Plugin
BatchClassVersion	This is the version number of the batch class under which the batch was processed. This is the value of the batch_class_version column in the batch_class table. Each batch in Ephesoft Transact is run under a batch class that is a single unit for all configurations and workflow definitions. A foreign key relation is established between the ID column of the batch_class table and the batch_class_id column in the batch_instance table.	Folder Import	IMPORT_BATCH_FOLDER_PLUGIN
BatchName	This is the value of the batch_name column in the batch_instance table.	Folder Import	IMPORT_BATCH_FOLDER_PLUGIN
BatchDescription	This is the batch instance description that is provided at the time of batch creation.	Folder Import	IMPORT_BATCH_FOLDER_PLUGIN
BatchPriority	This is the value of the batch_priority column in the batch_instance table. Priority can be a value between 1 to 100 with the lower number having higher priority. If not assigned using custom code the batch priority will be the priority from the batch class which is assigned when the batch class is created or imported.	Folder Import	IMPORT_BATCH_FOLDER_PLUGIN

Batch-Level Field Name	Description	Module	Plugin
BatchStatus	<p>This value specifies the current batch status. Possible values for this field are:</p> <ul style="list-style-type: none"> • New • Ready • Running • ReadyForReview • ReadyForValidation • Error • Finished 	All modules can modify this field	All plugins can modify this field
BatchReviewedBy	<p>This field cites users who performed batch review and enables administrators to audit users who are active in the batch instance. This node applies at the batch level and the individual page level.</p> <ul style="list-style-type: none"> • Batch level - This node indicates a single user who completed the batch. 	Page Process	SEARCH_CLASSIFICATION MULTIDIMENSIONAL_ CLASSIFICATION_PLUGIN
BatchValidatedBy	<p>This field cites users who performed validation and enables administrators to audit all users who are active in the batch instance, and this node applies at the batch level and the individual page level, with these parameters.</p> <ul style="list-style-type: none"> • Batch level - This node indicates a single user who completed the batch. 	Page Process	SEARCH_CLASSIFICATION MULTIDIMENSIONAL_ CLASSIFICATION_PLUGIN
BatchCreationDate	<p>This is the value of the creation_date column in the batch_class table. This is the date and time of when the batch was created in Transact.</p>	Folder Import	IMPORT_BATCH_FOLDER_PLUGIN

Batch-Level Field Name	Description	Module	Plugin
BatchLocalPath	This is the Transact system folder path where the batch instance folder will be available. This value will be the same across all batches in the system.	Folder Import	IMPORT_BATCH_FOLDER_PLUGIN
BatchSource	This value states which ingestion mechanism was used to import the batch into system. Possible values are: <ul style="list-style-type: none"> • Upload Batch • Web Scanner • Email Import • UNC Folder • Web Service • CMIS Import • Snapdoc 	Folder Import Module	IMPORT_BATCH_FOLDER
UNCFolderPath	This is the path where the source file for the batch is available. This is a unique path for each batch in the system.	Folder Import	IMPORT_BATCH_FOLDER_PLUGIN
ETextMode	The batch.xml will have an additional tag, ETextMode, at the root level to define the EText mode. Values may be Automatic, Always, and Never. The value will be populated based on the plugin setting for the following: <ul style="list-style-type: none"> • For Windows, use the EText Recostar Project property value in RECOSTAR_HOCR plugin of the Page Process module. • For Linux, use the Process PDFs as EText Files property value in OMNIPAGE_HOCR plugin of Page Process module. 	Page Process	RECOSTAR_HOCR OMNIPAGE_HOCR

Batch-Level Field Name	Description	Module	Plugin
DocumentClassification Types	This value specifies what classification type was used for the document assembly.	Assembly	DOCUMENT_ASSEMBLER

Document fields

The document fields in the batch.xml schema apply to all documents in the batch instance. Document fields exist at a higher level than document-level fields.

i For information on email document fields in the batch.xml schema, see [Email metadata in the batch.xml schema](#).

The following example illustrates the docField for a batch instance:

```
<xs:complexType name="docField">
  <xs:complexContent>
    <xs:extension base="field">
      <xs:sequence>
        <xs:element name="AlternateValues" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element minOccurs="0" maxOccurs="unbounded"
name="AlternateValue" type="field" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="PreviousValue" type="docField" minOccurs="0" maxOccurs="1" />
      </xs:sequence>
      <xs:element name="Category" type="xs:string" minOccurs="0" maxOccurs="1" />
      <xs:element name="hidden" type="xs:boolean" minOccurs="0" maxOccurs="1" />
      <xs:element name="widgetType" type="xs:string" minOccurs="0" maxOccurs="1" />
      <xs:element name="scriptEnabled" type="xs:boolean" minOccurs="0"
maxOccurs="1" />
      <xs:element name="Message" type="xs:string" minOccurs="0" maxOccurs="1" />
    </xs:sequence>
  </xs:extension>
</xs:complexContent>
```


```
</xs:complexType>
```

The following table lists and defines the fields contained in the docFieldsection of the batch.xml schema.

Document Field Name	Description	Module	Plugin
AlternateValues	<p>This field contains alternate values for a page-level field. This field stores alternative classification information with confidence levels. The type with the highest confidence value will be set in the page-level field value. All other possible types for a page will be present in alternative values. This tag will contain the LearnedFileName tag for all the alternate values. During classification, the default value for this field is to 5.</p>	Page Process	SEARCH_CLASSIFICATION MULTIDIMENSIONAL_ CLASSIFICATION_PLUGIN

Document-level fields

The following table lists document-level fields in the Transact batch.xml schema and XSD.

 For email document-level fields in the batch.xml schema, refer to [Email metadata in the batch.xml schema](#).


Document-Level Field Name	Description	Module(s)	Plugin(s)
Identifier	<p>This value identifies a document. The sequence for document numbering is DOC0, DOC1...DOCn</p> <p>Each file becomes a separate batch, and there is only one document in the XML after Folder Import. All pages will belong to this one document.</p> <p>The page to document grouping will change after the DOCUMENT_ASSEMBLER_PLUGIN within the Page Process module is executed. The pages may be grouped into multiple documents.</p>	<p>Folder Import</p> <p>Document Assembly</p>	<p>IMPORT_BATCH_FOLDER_PLUGIN</p> <p>DOCUMENT_ASSEMBLER_PLUGIN</p>

Document-Level Field Name	Description	Module(s)	Plugin(s)
Type	<p>This is the document type assigned to the document.</p> <p>Each file becomes a separate batch, so there will only be one document in the XML after Folder Import. All pages will belong to this one document and named Unknown. The page to document grouping will change after the DOCUMENT_ASSEMBLER_PLUGIN within the Page Process module is executed. The pages may be grouped into multiple documents. The document that Transact used to determine classification for all pages is assigned this tag. The document types that belong to the batch class assigned to the batch is available in the database table document_type (field - document_type_name). This table has a foreign key reference to the ID column of the batch_class table that associates documents to batch class.</p>	<p>Folder Import Document Assembly</p>	<p>IMPORT_BATCH_FOLDER_PLUGIN DOCUMENT_ASSEMBLER_PLUGIN</p>

Document-Level Field Name	Description	Module(s)	Plugin(s)
ExtractionType	<p>The possible values for the element are:</p> <ul style="list-style-type: none"> • System: When the index field is automatically extracted and the operator does not modify this value, <ExtractionType> is set to system. The original XY coordinates for the extracted values are retained. • Clicked: When the index field is automatically extracted and the operator changes this value by clicking on the full image displayed and extracting the OCR text, <ExtractionType> is set to clicked. The new XY coordinates from the image replace the original XY coordinates. These coordinates are retained in batch.xml. • Edited: When the index field is automatically extracted and the operator manually edits this value, the <ExtractionType> is set to edited. The original XY coordinates for the extracted values are retained. 		
Description	<p>This tag contains the corresponding document_type_description of the assigned document_type_name above. It is picked up from the database table document_type.</p>	<p>Folder Import Document Assembly</p>	<p>IMPORT_BATCH_FOLDER_PLUGIN DOCUMENT_ASSEMBLER_PLUGIN</p>
Size	<p>This element contains the document's multipage PDF file size in bytes. This field is only populated by the IBM_CM_PLUGIN.</p>	<p>Export Module</p>	<p>IBM_CM_PLUGIN</p>

Document-Level Field Name	Description	Module(s)	Plugin(s)
Confidence	This value is the confidence with which the document was assembled. If the confidence is greater than the minimum confidence threshold assigned to the document, then the document is not marked for operator review.	Folder Import	IMPORT_BATCH_FOLDER_PLUGIN DOCUMENT_ASSEMBLER_PLUGIN
OcrConfidenceThreshold	This field helps Transact to decide if the document should skip document review automatically when the classification score is higher than the threshold. The document confidence threshold is available in the table document_type (located in the field-min_confidence_threshold column). The best practice is to set the threshold so that false positives are minimized.	Document Assembly	DOCUMENT_ASSEMBLER_PLUGIN
OcrConfidence	This is the confidence value returned from the OCR engine used with Transact.		
CoordinatesList	This tag includes the <ExtractionType> element.		
ExtractionRuleID	The Rule ID allows batch class designers to identify which rule has been applied for the extraction of each index field. The Rule ID can be found as an attribute listed for a key-value extraction rule.	Extraction Module	KEY_VALUE_EXTRACTION

Document-Level Field Name	Description	Module(s)	Plugin(s)
Valid	This tag determines if the document would stop for Document Field Validation review. This applies only when data extraction is part of the batch class. The value of False indicates that the document has fields that need to stop for Document Field Validation review. The value of True indicates that all fields in the document were extracted with high confidence and need not stop for Document Field Validation review. The value is set to True after execution of REVIEW_DOCUMENT_PLUGIN if the extraction module is not configured in the batch class.	Document Assembly Review Document	DOCUMENT_ASSEMBLER_PLUGIN REVIEW_DOCUMENT_PLUGIN

Document-Level Field Name	Description	Module(s)	Plugin(s)
Reviewed	<p>This tag indicates that the document has passed through the REVIEW_DOCUMENT_PLUGIN. The value of False indicates that the document was assembled/classified with low confidence and needs to stop for document classification Review. The value of True indicates that the document was assembled/classified with high confidence and does not need to stop for document classification Review. The value is set to True after execution of REVIEW_DOCUMENT_PLUGIN.</p> <div data-bbox="553 953 826 1220" style="border: 1px solid #add8e6; padding: 5px; background-color: #e6f2ff;"> <p> Setting the value to False will not force the batch to stop in Review. To stop the batch, you will need to set the confidence score that is lower than the threshold.</p> </div>	Document Assembly Review Document	DOCUMENT_ASSEMBLER_PLUGIN REVIEW_DOCUMENT_PLUGIN
ReviewedBy	<p>This field lists users who performed reviews in the batch. This field enables administrators to audit users who are active in the batch instance. This node applies at the batch level and the individual page level:</p> <ul style="list-style-type: none"> • Page level: This node accommodates multiple users if multiple users processed pages in the batch. <p>See Case Study: ReviewedBy Node in Batch.xml.</p>	Page Process	SEARCH_CLASSIFICATION MULTIDIMENSIONAL_CLASSIFICATION_PLUGIN


Document-Level Field Name	Description	Module(s)	Plugin(s)
ValidatedBy	<p>This field lists users who performed validation for the batch. This node enables administrators to audit all users who are active in the batch instance, and this node applies at the batch level and the individual page level, with these parameters:</p> <ul style="list-style-type: none"> • Batch level: This node indicates a single user who completed the batch. • Page level: This node accommodates multiple users when multiple users processed pages in the batch. 	Page Process	SEARCH_CLASSIFICATION MULTIDIMENSIONAL_ CLASSIFICATION_PLUGIN
ErrorMessage	A string that contains an error message to be displayed on the Review and Validate screen corresponding to a document. The value of this tag can be set using a scripting plugin. Transact does not set a value for this field.	Review/Validation	Review/Validation
Document DisplayInfo	This field can be used to provide customized names to documents on the Review and Validate screen. The value of this tag can be set using a scripting plugin. Transact does not set a value for this field.	Review/Validation	Review/Validation
Document LevelFields	This field contains the parent node for all index fields inside the documents.	Extraction Module	All plugins inside extraction module can modify this field.
Pages	This field contains the parent node for all pages inside the document.	Document Assembly	DOCUMENT_ASSEMBLER


Document-Level Field Name	Description	Module(s)	Plugin(s)
DataTables	This is the root node for all the extracted table information in the document.	Extraction Module	TABLE_EXTRACTION
AutoSuggestedDataTables	All the auto extraction from documents is present under this node.	Extraction Module	AUTO_TABLE_EXTRACTION_PLUGIN
MultiPage TiffFile	This value contains the name of the multipage TIFF file created. The COPY_BATCH_XML plugin changes this name to the exact location while exporting the file to a user-defined location. Please note the batch.xml in the <code>ephesoft-system-folder</code> in the shared folders that still contain the file name.	Export Module	CREATEMULTIPAGE_FILES COPY_BATCH_XML
MultiPage PdfFile	This value contains the name of the multipage PDF file created. The COPY_BATCH_XML plugin changes this name to the exact location while exporting the file to a user-defined location. Please note the file name of the batch.xml contained in the shared folders of the <code>ephesoft-system-folder</code> .	Export Module	CREATEMULTIPAGE_FILES COPY_BATCH_XML
FinalMultiPage PdfFilePath	This element is present in the batch.xml in the <code>ephesoft-system-folder</code> and contains the absolute path of the exported multipage PDF document (by the COPY_BATCH_XML plugin). This value is different from MultiPagePdfFile which contains the file name only.	Export Module	COPY_BATCH_XML

Document-Level Field Name	Description	Module(s)	Plugin(s)
FinalMultiPage TiffFilePath	This element is present in the batch.xml in the <code>ephesoft-system-folder</code> and contains the absolute path of the exported multipage TIFF document by the COPY_BATCH_XML plugin. It is different from MultiPagePdfFile which contains the file name only.	Export Module	COPY_BATCH_XML

Page fields

Page fields in the batch.xml schema and XSD apply to all pages in the document. See the following table for the page fields in the Transact batch.xml schema and XSD.

 For email document-level fields in the batch.xml schema, refer to [Email metadata in the batch.xml schema](#).

Page Field Name	Description	Module	Plugin
Identifier	<p>This is the document identifier for a document. The sequence for document numbering is PG0, PG1...PGn.</p> <p> The IMPORT_BATCH_FOLDER_PLUGIN breaks up each page of the source PDF into individual TIFF files. Each TIFF file is a page in the XML file. The pages can be grouped as documents.</p>	Folder Import	IMPORT_BATCH_FOLDER_PLUGIN
OldFileName	This tag contains the name of the mapped individual TIFF file within the input folder for the batch. The input folder path is available in the tag UNCFolderPath under batch-level fields.	Folder Import	IMPORT_BATCH_FOLDER_PLUGIN

Page Field Name	Description	Module	Plugin
NewFileName	This tag contains the name of the mapped individual TIFF file within the Transact system folder. The Transact system folder path is available in the tag BatchLocalPath. The path to the batch instance folder is <code><BatchLocalPath>\<BatchInstanceIdentifier></code> . The name of the associated file to this page is a combination of the batch instance identifier and the page sequence.	Folder Import	IMPORT_BATCH_FOLDER_PLUGIN
SourceEmailID	This element links the page to the source email from which the page originated. It contains only the identification (ID) of the source email. This ID can be searched for in a separate section of the batch.xml which contains more details.	Folder Import Module	IMPORT_BATCH_FOLDER
SourceFileID	This element links the page to the source file which was originally placed in the watch folder. It only contains id of the source file. This id can be looked up in a separate section in the batch.xml which contains more details.	Folder Import Module	IMPORT_BATCH_FOLDER
PageLevelFields	This value contains the classification information from different configured plugins. The values in this section are used while assembling pages into documents.	Document Assembly	DOCUMENT_ASSEMBLER

Page Field Name	Description	Module	Plugin
HocrFileName	The RECOSTAR_HOCR_GENERATION_PLUGIN (for Windows) or the OMNIPAGE_HOCR (for Linux) extract the contents of each page (individual TIFF). The contents are stored in an XML file which is located in the batch instance folder (<BatchLocalPath>\<BatchInstanceIdentifier>). This tag stores the name of the HOCR XML file for the corresponding page.	Page Process	RECOSTAR_HOCR_GENERATION_PLUGIN
ThumbnailFileName	This tag stores the name of the corresponding thumbnail for the page. The IMAGE_PROCESS_CREATE_THUMBNAIIS_PLUGIN is used to create thumbnail images of the batch images. These thumbnails are displayed in the Review and Validate screen, where pages in the documents are shown as thumbnails under the document name. The thumbnails are stored in the batch instance folder (<BatchLocalPath>\<BatchInstanceIdentifier>).	Page Process	IMAGE_PROCESS_CREATE_THUMBNAIIS_PLUGIN
ComparisonThumbnailFileName	This value contains the name of the thumbnail file which can be used by the CLASSIFY_IMAGES plugin. This element will be present only when the Create Compare Thumbnail Switch is on in the CREATE_THUMBNAIIS plugin.	Page Process	CREATE_THUMBNAIIS

Page Field Name	Description	Module	Plugin
DisplayFileName	<p>The IMAGE_PROCESS_CREATE_DISPLAY_IMAGE_PLUGIN performs the functionality of creating the display PNG files for the images being processed. This plugin takes all the images and creates PNG files for the corresponding pages and is displayed on the Review and Validate screens. The display images are stored in the batch instance folder (<BatchLocalPath> \<BatchInstanceIdentifier>).</p> <p>This tag stores the name of the corresponding display image for the page.</p>	Page Process	IMAGE_PROCESS_CREATE_DISPLAY_IMAGE_PLUGIN
OCRInputFileName	<p>This tag stores the name of the file that was used by the RECOSTAR_HOCR_GENERATION_PLUGIN to extract the contents of the page. The image will be the corresponding individual TIFF for the page available in the batch instance folder (<BatchLocalPath> \<BatchInstanceIdentifier>).</p>	Page Process	RECOSTAR_HOCR_GENERATION_PLUGIN
Direction	<p>This field indicates the direction of a rotated document.</p>	Folder Import	IMPORT_BATCH_FOLDER_PLUGIN
IsRotated	<p>This field indicates whether a document is rotated on the Review and Validate screen.</p>	Folder Import	IMPORT_BATCH_FOLDER_PLUGIN

Page Field Name	Description	Module	Plugin
TreatAsEText	For a PDF, the batch.xml is updated for each page that is EText compatible or not EText compatible. The element at the page level is <TreatAsEText> with values of true or false. The <TreatAsEText> element will not display for the pages having an input file other than PDF in UNC.	Page Process	RECOSTAR_HOCR OMNIPAGE_HOCR
ImprintedString	Read the imprinted string from serialized file for each image. If the imprinter was enabled during scanning, then add this string as a page-level field in batch.xml. This will be used during rescan. Insert the functionality on the Review and Validate screen.	Folder Import	IMPORT_BATCH_FOLDER
IsBlank	IsBlank under the Page tag is used if there is no HOCR content associated with the page (i.e., if there is a blank HOCR XML associated with the page or image).	Page Process	RECOSTAR_HOCR OMNIPAGE_HOCR

Page-level fields

Page-level fields apply to each document page in the batch.xml file. See the following table for the page-level fields in the Transact batch.xml schema and XSD.

i For email document-level fields in the batch.xml schema, refer to [Email metadata in the batch.xml schema](#).

Page-Level Field Name	Description	Module	Plugins
Name	This tag contains the name of the classification method used to classify this page.	Page Process	SEARCH_CLASSIFICATION MULTIDIMENSIONAL_ CLASSIFICATION_PLUGIN

Page-Level Field Name	Description	Module	Plugins
Value	Each document type within the batch class is subdivided into pages (first, middle, and last). This tag holds the document page for which this page was classified.	Page Process	SEARCH_CLASSIFICATION MULTIDIMENSIONAL_ CLASSIFICATION_PLUGIN
Type	This tag is used in bar code classification only where it keeps the information about the bar code type.	Page Process	SEARCH_CLASSIFICATION MULTIDIMENSIONAL_ CLASSIFICATION_PLUGIN
Confidence	This tag holds the confidence score with which the page was classified. The DOCUMENT_ASSEMBLER plugin uses this confidence score during document assembly once the workflow enters the Document Assembly module.	Page Process	SEARCH_CLASSIFICATION MULTIDIMENSIONAL_ CLASSIFICATION_PLUGIN
LearnedFileName	This tag holds the name of the lucene-search-classification-sample matched against this page/image.	Page Process	SEARCH_CLASSIFICATION MULTIDIMENSIONAL_ CLASSIFICATION_PLUGIN

Email metadata in the batch.xml schema

Transact supports email-specific metadata passing through to the batch.xml schema. For additional information about accessing and ingesting email header information for batch instances and how that metadata is passed through to the batch.xml, see "Accessing email headers in the batch.xml schema" in the *Ephesoft Transact Help*.

Case studies for batch.xml

The following case studies describe configuration steps that may be helpful to you when customizing your Transact environment:

- Accessing the batch.xml file case study.
- Batch-level case study that uses the BatchReviewedBy and BatchValidatedBy fields in which there are three active users.
- Document-level case study that shows the ReviewedBy and ValidatedBy Nodes in a sample batch.xml.

Case study 1: Accessing batch.xml

Follow these steps to access batch.xml for your deployment of Transact.

1. Open Transact, log in as Administrator and open the **Batch Class Management** screen.
2. Open a batch class that contains a fully configured **Export** module, and in which at least one batch instance has been processed.
3. In the Export module, select the **COPY_BATCH_XML** plugin. The Plugin Configuration screen for this plugin appears on the right.
This screen displays the path to the batch.xml file in the field titled **Batch XML Export Folder Location** field.
4. Access the batch.xml file, navigate to this path on the Transact server. In the subfolder named `final-drop-folder`, there will appear another subfolder that is named for the batch instance.
The batch.xml file contained in this subfolder is also named according to the batch instance in which it was created.
5. Open the batch.xml file with Notepad++ or a similar text editor by right-clicking the file, selecting **Open With**, and choosing an application.

Client-side scripting

There are three JavaScript methods to handle different events on the Web Scanner and Upload Batch screens.

- Selecting a Batch Class when UI renders.
-
-
-
-

Selecting a batch class when the user interface renders

Description: A method is called when the UI renders. A String Identifier is expected as a return out of it. If the method returns null , then default view would be shown, else the Batch Class with Identifier will be selected by default.

Purpose: This method provides additional functionality to auto select batch class on Upload batch and web scanner screen, when screen is opened.

Prepopulating Field Values

Description:Method is called before displaying the pop up for the Batch Class Fields. Returned Values is e prepopulated values for the fields.

Purpose: This method provides functionality to pre populating batch class fields depending on batch class identifier.

Calling JavaScript methods when field value changes

Description:Handled the event of Batch Class Fields Value change and provided an interface to call a function to expect a JSON as a return, which sets the returned values to all the other fields.

Purpose: To set other batch class fields based on value of a batch class field.

Following are the new methods have been added to the utility.js file present at Ephesoft/ Application path for the required functionality.

```

/*
 * Method to pre populate batch class in web scanner and upload batch.
 */
function getBatchClass() {
    return null;
}

/*
 * Method to pre populate batch class fields on basis of batch class identifier
 * and description in web scanner and upload batch.
 */
function getBatchClassFields(batchClassIdentifier, batchClassDescription) {
    var bcf = '[{"key":"field1","value":"value1"}, {"key":"field2","value":"value2"}, {"key":"field3","value":"value3"}]';
    return bcf;
}

/*
 * Method to pre populate batch class fields on basis of batch class identifier,
 * batch class field, value typed and description in web scanner and upload
 * batch.
 */
function getBatchClassFieldsWhileFieldValueChange(batchClassIdentifier,
    batchClassDescription, key, value) {
    var bcf = '[{"key":"field1","value":"value1"}, {"key":"field2","value":"value2"}, {"key":"field3","value":"value3"}]';
    return bcf;
}

```

The following lists the JavaScript methods.

getBatchClass

Function: Returns the batch class identifier that needs to be set on the Upload Batch and Web Scanner screens.

Parameters: None

Example: Sets BC8 on Upload Batch and Web Scanner:

```

function getBatchClass(){
    return "BC8";
}

```

getBatchClassFields

Function: Pre-populate batch class fields when the Fields button is clicked on the Upload Batch and Web Scanner screens. You can configure different BCF keys and values on the basis of batch class identifiers and descriptions.

Parameters: batchClassIdentifier, batchClassDescription

Example:

```

function getBatchClassFields(batchClassIdentifier, batchClassDescription) {
    var bcf=' [{"key":"field1","value":"value1"}, {"key":"field2","value":"value2"}, {"key":"field3","value":"value3"}]';
    return bcf;
}

```


getBatchClassFieldsWhileFieldValueChange

Function: Populate batch class fields when the field changes. You can configure different values to be set in BCF on the basis of batch class identifier, batch class description, key, and value.

Parameters: batchClassIdentifier, batchClassDescription, key, value

```
function
getBatchClassFieldsWhileFieldValueChange(batchClassIdentifier, batchClassDescription, key, value)
{
    var bcf=[{"key":"field1","value":"value1"}, {"key":"field2","value":"value2"},
{"key":"field3","value":"value3"}];
    return bcf;
}
```

Error causes for default scripts

The default scripts contain descriptions of errors that occur. If an exception is thrown at any level of the batch processing workflow, the error appears on the **Batch Instance Management** screen under **Batch Execution Details** in the **Error Cause** field.

Error comments are included in all default scripts except ScriptPageProcessing and ScriptValidation. Below is the list of error messages found in the Transact default scripts:

Default Script Error Message	Scripts
Document doesn't exist.	ScriptAddNewTable
Input document is null.	ScriptAddNewTable ScriptAutomaticValidation ScriptDocumentAssembler ScriptExport ScriptExtraction ScriptFieldValueChange ScriptFunctionKey ScriptNewTableRowInsert ScriptTableCellValueChange
Unable to find the local folder path in batch xml file.	ScriptAddNewTable ScriptAutomaticValidation ScriptDocumentAssembler ScriptExport ScriptExtraction ScriptFieldValueChange ScriptFunctionKey ScriptNewTableRowInsert ScriptTableCellValueChange

Default Script Error Message	Scripts
Unable to find the batch instance ID in batch xml file.	ScriptAddNewTable ScriptAutomaticValidation ScriptDocumentAssembler ScriptExport ScriptExtraction ScriptFieldValueChange ScriptFunctionKey ScriptNewTableRowInsert ScriptTableCellValueChange
Unable to read the zip switch value. Taking default value as true. Exception thrown is:	ScriptAddNewTable ScriptAutomaticValidation ScriptDocumentAssembler ScriptExport ScriptExtraction ScriptFieldValueChange ScriptFunctionKey ScriptNewTableRowInsert ScriptTableCellValueChange
***** Error occurred in scripts.	ScriptAddNewTable ScriptAutomaticValidation ScriptDocumentAssembler ScriptExport ScriptExtraction ScriptFieldValueChange ScriptFunctionKey ScriptNewTableRowInsert ScriptPageProcessing ScriptTableCellValueChange ScriptValidation

We recommend leaving error messages in default scripts unchanged. If required, you can customize the entire script and provide any error message as needed. Default scripts are in the batch class folder created at the time of batch class configuration (`SharedFolders\<<Batch Class>\scripts`).

Enable logging for custom scripts

Transact has a built-in logging functionality. When setting `System.out.println` in the scripts, Transact always print statements to the logs, which can cause the logs to grow large very quickly. By using the built in `log4j` logger, the scripts will only print to the logs based on the logging level specified in the `log4j.xml` file. Replace the messages to match your specific logging alerts.

Use the sample below to create logs for the script to print to the stdout and dcma-all log files.

```

public Object execute(Document document, String methodName, String docIdentifier) {
    Exception exception = null;
    try {
        LOGGER.info("***** Inside ExportScript scripts.");

        LOGGER.info("***** Start execution of the ExportScript scripts.");

        if (null == document) {
            LOGGER.error("Input document is null.");
            return null;
        }

        LOGGER.info("***** End execution of the ScriptExport scripts.");
    } catch (Exception e) {

        LOGGER.error("***** Error occurred in scripts." + e.getMessage());

        exception = e;
    }
    return null;
}

```

JDOM script configuration

Update the `dcma-scripting-plugin.properties` in `<Transact_INSTALL_DIR>\EphesoftApplication\WEB-INF\classes\META-INF\dcma-scripting-plugin`. Locate this setting:

```
script.parser_type= jdom
```

If anything other than `jdom` is specified for `script.parser_type`, the `IScript` parser will be used and Scripts using `IScript` will be executed.

Sample scripts to compare IScript and JDOM

This simple example for converting `IScript` scripts into `JDOM` scripts. Changes to be made to convert the present script to `JDOM`.

1. Change import statements from `IScript` to `JDOM`:

Change IScript statements	To JDOM statements
<pre>import org.w3c.IScript.Document; import org.w3c.IScript.Node; import org.w3c.IScript.NodeList; import com.ephesoft.dcma.script.IScripts;</pre>	<pre>import org.jdom.Document; import org.jdom.Element; import org.jdom.output.XMLOutputter; import com.ephesoft.dcma.script.IJDomScript;</pre>
<pre>import com.ephesoft.dcma.script.IScripts;</pre>	<pre>import com.ephesoft.dcma.script.IJDomScript;</pre>

2. Change the implements statement. `IJDomScript` is the interface for running the `JDOM` Scripts. Each scripts should implements `IJDomScript` for running the customize script using `JDOM`.

Change IScript statement	To JDOM statement
<pre>public class ScriptAutomaticValidation implements IScripts {</pre>	<pre>public class ScriptAutomaticValidation implements IJDomScript {</pre>

3. Make API changes from IScript to JDOM.

Getting the list document object in JDOM is different from IScript. In JDOM, we need to iterate each node for getting the child element list because JDOM does not access the child tag directly without accessing the parent tag of that child tag.

- Getting page nodes:

IScript

```
// Getting directly PAGES tag without accessing its parent in batch.xmlNodeList
documentList = document.getElementsByTagName(PAGES);
```

JDOM

```
// Getting PAGES tag with accessing its parent in batch.xml
// Getting first child of root nodeElement documents =
document.getRootElement().getChild(DOCUMENTS);
// Getting first child of DOCUMENTS node

List documentList = documents.getChildren(DOCUMENT); for (int documentIndex =
0; documentIndex < documentList.size(); documentIndex++) { Element document
= (Element) documentList.get(documentIndex); Element pages = (Element)
document.getChild(PAGES); }
```

- Getting text content from element:

IScript

```
String elementValue = element.getTextContent();
```

JDOM

```
String elementValue = element.getText();
```

- Setting text context in element:

IScript

```
element.setTextContent("String Data");
```

JDOM

```
element.setText("String Data");
```

- To get a child node for any particular node:

IScript

```
Element variableName = (Element)
parentElement.getChild("Name_of_child");
```

JDOM

```
Element variableName = (Element)
parentElement.getChild("Name_of_child");
```

- To get the textual content of the named child element:

IScript

```
String name = parentElement.getElementsByTagName(" Name
").item(0).getTextContent();
```

JDOM

```
Element.setText("Text_to_be_put_as_name");
```

- To create a new child node in the parent:

IScript

```
Element childElement =
document.createElement("Child");parentElement.appendChild(childElement);Here
document is the argument passed to the method.
```

Example:

```
"execute(Document document, String fieldName, String docIdentifier)"
```

JDOM

```
Element newElement = new Element("Name_of_element");
parentElement.addContent( newElement);
```

Implement .zip functionality to older scripts

If your older scripts do not have the ability to create .zip files, make the following changes to each script.

1. Add the following import statements to the script java class file:

```
import java.util.zip.ZipEntry;
import java.util.zip.ZipOutputStream;
import java.io.File;
import java.io.FileWriter;
import java.io.OutputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.FileNotFoundException;
import java.util.List;
```

2. Add the following property after the import statements:

```
private static String ZIP_FILE_EXT = ".zip";
```

3. Replace the entire writeToXML method in each of the scripts with the following:

```
private void writeToXML(Document document) {
    String batchLocalPath = null;
    Element batchLocalPathElement =
document.getRootElement().getChild(BATCH_LOCAL_PATH);
    if (null != batchLocalPathElement) {
        batchLocalPath = batchLocalPathElement.getText();
        if (null == batchLocalPath) {
            System.err.println("Unable to find the local folder path in batch xml
file.");
            return;
        }
    }
    String batchInstanceID = null;
    Element batchInstanceIDElement =
document.getRootElement().getChild(BATCH_INSTANCE_ID);
    if (null != batchInstanceIDElement) {
        batchInstanceID = batchInstanceIDElement.getText();
    }
}
```

```

    }
    if (null == batchInstanceID) {
        System.err.println("Unable to find the batch instance ID in batch xml
file.");
        return;
    }
    String batchXMLPath = batchLocalPath.trim() + File.separator +
batchInstanceID + File.separator + batchInstanceID+ EXT_BATCH_XML_FILE;
    String batchXMLZipPath = batchXMLPath + ZIP_FILE_EXT;
    System.out.println("batchXMLZipPath*****" + batchXMLZipPath);
    OutputStream outputStream = null;
    File zipFile = new File(batchXMLZipPath);
    FileWriter writer = null;
    XMLOutputter out = new XMLOutputter();
    try {
        if (zipFile.exists()) {
            System.out.println("Found the batch xml zip file.");
            outputStream = getOutputStreamFromZip(batchXMLPath, batchInstanceID
+ EXT_BATCH_XML_FILE);
            out.output(document, outputStream);
        }
        else {
            writer = new java.io.FileWriter(batchXMLPath);
            out.output(document, writer);
            writer.flush();
            writer.close();
        }
    }
    catch (Exception e) {
        System.err.println(e.getMessage());
    }
    finally {
        if (outputStream != null) {
            try {
                outputStream.close();
            }
            catch (IOException e) {
            }
        }
    }
}
}
}

```

4. Add the following method to the script java class file, as it is required by the new writeToXml() function:

```

public static OutputStream getOutputStreamFromZip(final String zipName, final
String fileName) throws FileNotFoundException, IOException {
    ZipOutputStream stream = null;
    stream = new ZipOutputStream(new FileOutputStream(new File(zipName +
ZIP_FILE_EXT)));
    ZipEntry zipEntry = new ZipEntry(fileName);
    stream.putNextEntry(zipEntry);
    return stream;
}

```

Testing scripts

We recommend testing custom scripts outside of Transact. This speeds up development because you do not need to wait for batches to process, and it does not affect a Transact installation in production. You can test scripts in your choice of IDE, such as NetBeans and Eclipse.

Follow these steps to test the script.

1. In your IDE, open the Transact custom script.
2. Write a main method in the script just before the execute method in the script.
This main method enables the script to be run outside of Transact. The filePath parameter must be set to the path of the batch.xml file, as shown in this example.


```
public static void main(String args[]){
    //Define a path to the Batch XML. In this case the BI19_Batch.xml
    was extracted and placed into a folder that was created call scriptdev
    String filePath = "C:\Ephesoft\SharedFolders\ephesoft-system-
    folder\BI19\Scriptdev\BI19_batch.xml";
    DocumentBuilderFactory dbfac =
    DocumentBuilderFactory.newInstance();
    try {
        SAXBuilder sb = new SAXBuilder();
        Document doc = sb.build(filePath);
        //Note that the name of the script is being used for
        the execute
        ScriptExtraction se = new ScriptExtraction();
        se.execute(doc, null, null);
    } catch (Exception x) {
    }
}
```

3. If the script requires Transact-specific classes, add the .jar files in the class path located in <Transact_Home>/WEB-INF/lib/.
4. Run the script in your IDE.
5. When you are finished, comment out or delete the main method you added in step 2.

Trigger field value change script for table data fields

This feature helps users run custom scripts while changing the table data on the validation screen. The system triggers a validation script if you change any table field. The system control defaults to the first invalid entry on the table view if the script is on. This feature works the same way as the field value change script works for document-level fields. Follow these steps


1. Go to the **Batch Class Management** screen.
2. Select the batch class you want to modify or create a new one.
3. Create any tables for the batch class as needed.
4. In the table listing, select the **Table Cell Value Change Script** value for the table.
5. In **Validate Document** plugin, set **Table Cell Value Change Script Switch** to **ON**.
6. Configure the table columns and table extraction rules.
7. Apply and deploy the newly added configuration.
8. Modify the script file, ScriptTableCellValueChange.java, by doing the following:
 - a. Go to Ephesoft\SharedFolders\Batch_Identifier.
 - b. Open the script.

 The script is bound to the table.

- c. Configure the script as required.

The following information appears in the script:

- Updated document object
- Table Name
- Document Identifier
- Table Column Name
- Row index(int)

 You can use the Ctrl+[(open bracket) shortcut to enable or disable the script.

You are now ready to execute the batch.

9. Run the batch once.
10. At the top of the **Validation** screen, click **Table**.
The **Table View** screen appears.
11. In the **Table View** screen, change the field value and press Tab.
The following should happen:
 - In the image, the application highlights the first invalid field on the table.
 - Any other changed fields are also highlighted.

Chapter 5

Transact Web Services API

Transact's OpenAPI-Compliant web services provide a simple method for real-time integration and exposure of Ephesoft Transact capabilities to external applications, allowing developers to embed advanced capture features in their own solutions without having to display the Transact user interface.

Transact provides over 60 Transact web services to perform advanced capture actions in your custom applications. These web services perform tasks that range from simple actions like determining which batch instances can be seen by an individual user, to performing complete OCR, classification, or extraction procedures on a collection of documents.

Transact web services support the self-documenting feature of the OpenAPI Specification v2.4.0 (formerly known as the Swagger Specification). This means that tools like Swagger UI can be used to browse the collection of web services and view detailed documentation for each web service. The Swagger UI tool is bundled with Ephesoft Transact, and can be accessed by navigating to the following URL in your Transact instance:

```
http://<server_name>:<port>/dcma/rest/swagger-ui.html
```

Navigating to this URL will take the user to the Transact Web Services Explorer. This interface lists all Transact web services on a single page, allowing both seasoned developers and "citizen developers" to browse and navigate through the Transact web services available within their server.

This interface also allows the user to review detailed information about each web service. Clicking a web service in the Web Services Explorer allows you to drill further into the web service to view additional details, as shown below.

The overall Transact OpenAPI-compliant web services definition file for all API endpoints is represented in JSON format. This definition file can be found at the following location in your Transact instance:

```
http://<server_name>:<port>/dcma/rest/v2/api-docs.json
```

Transact Web Services optimized for workflow engines

All Transact web services support the self-documenting capabilities of the OpenAPI Specification, but only two Transact web services, `v2/ocrClassifyExtract` and `v2/ocrClassifyExtractBase64`, provide simplified JSON responses, allowing them to be embedded into workflow engines like Nintex and Microsoft Flow. These minimized JSON responses have a smaller footprint and improved interoperability. An example of this minimized JSON response is shown below.


```
1 {
2   "Type": "UsInvoice",
3   "Confidence": 47.84,
4   "DocumentLevelField": [
5     {
6       "Name": "PONumber",
7       "Value": "234 2005012345",
8       "Confidence": 30,
9       "OcrConfidenceThreshold": 90,
10      "OcrConfidence": 100
11    },
12    {
13      "Name": "City",
14      "Value": "Irvine",
15      "Confidence": 100,
16      "OcrConfidenceThreshold": 90,
17      "OcrConfidence": 98
18    }
19  ]
20 }
```

These two web services have been optimized specifically for advanced workflow engines. See the following links for more information about these two web services:

- [v2/ocrClassifyExtract](#)
- [v2/ocrClassifyExtractBase64](#)

Because they support minimized JSON responses, these two web services can be fully integrated into advanced workflow engines such as Nintex or Microsoft Flow. These tools allow users to develop complex workflows that can integrate different products and services together, enabling decision-making processes with or without human intervention.

For example, a workflow can easily be constructed to monitor Box to look for any new invoices created in the `Incoming` folder. When a new invoice is created in that folder, the `v2/ocrClassifyExtract` web service can extract the invoice amount from the invoice and feed that data into a decision tree. Invoices with amounts of \$1000 or less can be approved automatically and routed to a `Processed` folder in Box. Invoices with amounts greater than \$1000 can trigger a notification for an employee to review and approve the invoice before releasing it to the `Processed` folder.

 Currently we support only basic authentication for Transact web services (base 64 encoded user name and password). Token-based authentication usually used with SAML, OAuth, and other Single Sign-On authentication frameworks are not supported.

Web service definitions and code samples

The Transact Web Services API provides a simple method for real-time integration and exposure of Transact processes to external applications. This allows developers to embed and employ advanced capture capabilities and technologies in content management systems and other workflows.

Transact web services support the OpenAPI standard, and the Transact Web Services Explorer uses the Swagger interface to enable documentation and testing for Transact web services. Web services that return XML responses can be tested directly in the Swagger interface, but due to a limitation of the Swagger interface, web services that return non-XML responses cannot. See the Swagger/OpenAPI Capabilities section of each web service for information about whether a web service supports being tested directly in the Swagger interface or not. All Transact web services, including those that cannot be tested in the Swagger interface, can still be tested in other web service testing tools, such as Postman.

This section describes how to provide [authentication for web services](#). It also includes a brief description of each web service followed by a code sample showing how the web service can be used. All code samples are written in Java and utilize the Apache Commons HttpClient library.


Web services are grouped into the following sections:

- [Batch class management web services](#)
- [Batch instance management web services](#)
- [Batch instance processing web services](#)
- [Image processing web services](#)
- [Reporting web services](#)

Authentication for web services

When Transact web services are used in an API development tool like Postman or Swagger, basic authentication must be provided in the http request header. For example:

```
Authorization:Basic ephesoft:demo
```

 Encode the user name and password in Base64.

When Transact web services are used in Java code, client call authentication can be handled like this:

```
Credentials defaultcreds = new UsernamePasswordCredentials("username", "password");  
client.getState().setCredentials(new AuthScope("serverName", 8080), defaultcreds);
```

Batch class management web services

copyBatchClass

This web service creates a copy of an existing batch class.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/batchClass/copyBatchClass

Sample client code using Apache Commons HttpClient

```
private static void createBatchClass() {
    HttpClient client = new HttpClient();
    String url = "http://{serverName}:{port}/dcma/rest/batchClass/copyBatchClass";
    PostMethod mPost = new PostMethod(url);
    // Adding Input XML file for processing
    File file = new File("C:\\sample\\Input.xml");
    Part[] parts = new Part[1];
    try {
        parts[0] = newFilePart(file.getName(), file);
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            String responseBody = mPost.getResponseBodyAsString();
            // Generating result as responseBody.
            System.out.println(statusCode + " *** " + responseBody);
        }
        else if(statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(mPost.getResponseBodyAsString());
        }
    } catch (FileNotFoundException e) {
        System.err.println("File not found for processing.");
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (mPost != null) {
            mPost.releaseConnection();
        }
    }
}
```

copyDocumentType

This web service copies an existing document type to create a new document type in the same batch class.

Request Method

POST

Web Service URL`http://<serverName>:<port>/dcma/rest/batchClass/copyDocumentType`**Sample client code using Apache Commons HttpClient**

```
private static void copyDocumentType() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/batchClass/copyDocumentType";
    PostMethod mPost = new PostMethod(url);
    // Adding Input XML file for processing
    File file = new File("C:\\sample\\Input.xml");
    Part[] parts = new Part[1];
    try {
        parts[0] = new FilePart(file.getName(), file);
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            String responseBody = mPost.getResponseBodyAsString();
            // Generating result as responseBody.
            System.out.println(statusCode + " *** " + responseBody);
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(mPost.getResponseBodyAsString());
        }
    } catch (FileNotFoundException e) {
        System.err.println("File not found for processing.");
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (mPost != null) {
            mPost.releaseConnection();
        }
    }
}
```

documentTypeCreator

This web service creates a new document type.

Request Method

POST

Web Service URL`http://<serverName>:<port>/dcma/rest/batchClass/documentTypeCreator`**Sample client code using Apache Commons HttpClient**

```
private static void createDocumentType() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/batchClass/documentTypeCreator";
    PostMethod mPost = new PostMethod(url);
```

```
// Adding Input XML file for processing
File file = new File("C: \\sample\\ Input.xml");
Part[] parts = new Part[1];
try {
    parts[0] = new FilePart(file.getName(), file);
    MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
    mPost.setRequestEntity(entity);
    int statusCode = client.executeMethod(mPost);
    if (statusCode == 200) {
        System.out.println("Web service executed successfully.");
        String responseBody = mPost.getResponseBodyAsString();
        // Generating result as responseBody.
        System.out.println(statusCode + " * * * "+responseBody);
    }
    else if(statusCode == 403) {
        System.out.println("Invalid username / password.");
    } else {
        System.out.println(mPost.getResponseBodyAsString());
    }
} catch (FileNotFoundException e) {
    System.err.println("File not found
for processing.");
} catch (HttpException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (mPost != null) {
        mPost.releaseConnection();
    }
}
}
```

exportBatchClass

This web service exports a batch class.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/exportBatchClass

Sample client code using Apache Commons HttpClient

```
private static void exportBatchClass() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/exportBatchClass";
    PostMethod mPost = new PostMethod(url);
    mPost.addParameter("identifier", "BC1");
    mPost.addParameter("lucene-search-classification-sample", "true");
    mPost.addParameter("image-classification-sample", "false");
    int statusCode;
    try {
        statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            System.out.println("Batch class exported successfully");
            InputStream in = mPost.getResponseBodyAsStream();
            File f = new File("C:\\sample\\serverOutput.zip");
            FileOutputStream fos = new FileOutputStream(f);
        }
    }
}
```

```
        try {
            byte[] buf = newbyte[1024];
            int len = in .read(buf);
            while (len > 0) {
                fos.write(buf, 0, len);
                len = in .read(buf);
            }
        } finally {
            if (fos != null) {
                fos.close();
            }
        }
    }
} else if(statusCode == 403) {
    System.out.println("Invalid username/password.");
} else {
    System.out.println(mPost.getResponseAsString());
}
} catch (HttpException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (mPost != null) {
        mPost.releaseConnection();
    }
}
}
```

getAllModulesWorkflowNameByBatchClass

This web service returns the module names and module workflow names for the specified batch class.

Request Method

GET

Web Service URL

http://<serverName>:<port>/dcma/rest/getAllModulesWorkflowNameByBatchClass/
<batchClassIdentifier>

Sample client code using Apache Commons HttpClient

```
private static void getAllModulesWorkflowNameByBatchClass() {
    HttpClient client = new HttpClient();
    // URL path to be hit for getting the module workflow name of the specified batch
    class identifier
    String url = "http://localhost:8080/dcma/rest/
getAllModulesWorkflowNameByBatchClass/BC1";
    GetMethod getMethod = new GetMethod(url);
    int statusCode;
    try {
        statusCode = client.executeMethod(getMethod);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            String responseBody = getMethod.getResponseAsString();
            System.out.println(statusCode + " *** " + responseBody);
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {

```

```
        System.out.println(getMethod.getResponseBodyAsString());
    }
} catch (HttpException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (getMethod != null) {
        getMethod.releaseConnection();
    }
}
}
```

getBatchClassFields

This web service returns the batch class fields of the supplied batch class identifier. The batch class fields are only returned if the batch class is accessible to the user.

Request Method

GET

Web Service URL

http://<serverName>:<port>/dcma/rest/getBatchClassFields/
<batchClassIdentifier>

getBatchClassForRole

This web service retrieves a list of all batch classes accessible to the specified user role.

Request Method

GET

Web Service URL

http://<serverName>:<port>/dcma/rest/getBatchClassForRole/<role>

Sample client code using Apache Commons HttpClient

```
private static void getBatchClassForRole() {
    HttpClient client = new HttpClient();
    // URL path to be hit for getting the batch class list having accessed by the role
    specified.
    String url = "http://localhost:8080/dcma/rest/getBatchClassForRole/admin";
    GetMethod getMethod = new GetMethod(url);
    int statusCode;
    try {
        statusCode = client.executeMethod(getMethod);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            String responseBody = getMethod.getResponseBodyAsString();
            System.out.println(statusCode + " *** " + responseBody);
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(getMethod.getResponseBodyAsString());
        }
    } catch (HttpException e) {
        e.printStackTrace();
    }
}
```



```
} catch (IOException e) {  
    e.printStackTrace();  
} finally {  
    if (getMethod != null) {  
        getMethod.releaseConnection();  
    }  
}  
}
```

getBatchClassList

This web service returns a list of all batch classes accessible by the authenticated user.

Request Method

GET

Web Service URL

http://<serverName>:<port>/dcma/rest/getBatchClassList

Sample client code using Apache Commons HttpClient

```
private static void getBatchClassList() {  
    HttpClient client = new HttpClient();  
    String url = "http://localhost:8080/dcma/rest/getBatchClassList";  
    GetMethod mGet = new GetMethod(url);  
    int statusCode;  
    try {  
        statusCode = client.executeMethod(mGet);  
        if (statusCode == 200) {  
            System.out.println("Web service executed successfully.");  
            String responseBody = mGet.getResponseBodyAsString();  
            System.out.println(statusCode + " *** " + responseBody);  
        }  
        else if(statusCode == 403) {  
            System.out.println("Invalid username/password.");  
        } else {  
            System.out.println(mGet.getResponseBodyAsString());  
        }  
    } catch (HttpException e) {  
        e.printStackTrace();  
    } catch (IOException e) {  
        e.printStackTrace();  
    } finally {  
        if (mGet != null) {  
            mGet.releaseConnection();  
        }  
    }  
}
```

getRoles

This web service retrieves all roles for a given batch class.

Request Method

GET

Web Service URL

http://<serverName>:<port>/dcma/rest/getRoles/<batchClassIdentifier>

Sample client code using Apache Commons HttpClient

```
private static void getRoles() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/getRoles/BC1";
    GetMethod mGet = new GetMethod(url);
    int statusCode;
    try {
        statusCode = client.executeMethod(mGet);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            String responseBody = mGet.getResponseBodyAsString();
            System.out.println(statusCode + " *** " + responseBody);
        }
        else if(statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(mGet.getResponseBodyAsString());
        }
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (mGet != null) {
            mGet.releaseConnection();
        }
    }
}
```

importBatchClass

This web service imports a batch class into Transact.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/importBatchClass

Sample client code using Apache Commons HttpClient

```
private static void importBatchClass() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/importBatchClass";
    PostMethod mPost = new PostMethod(url);
    mPost.setDoAuthentication(true);
    // Input XML for adding parameter.
    File file1 = new File("C:\\sample\\importbatchclass.xml");
    // Input zip file for importing batch class.
    File file2 = new File("C:\\sample\\BC1_050712_1714.zip");
    Part[] parts = new Part[2];
    try {
        parts[0] = new FilePart(file1.getName(), file1);
        parts[1] = new FilePart(file2.getName(), file2);
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
    }
```

```
int statusCode = client.executeMethod(mPost);
if (statusCode == 200) {
    System.out.println("Batch class imported successfully");
} else if (statusCode == 403) {
    System.out.println("Invalid username/password.");
} else {
    System.out.println(mPost.getResponseAsString());
}
} catch (FileNotFoundException e) {
    System.out.println("File not found for processing.");
} catch (HttpException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (mPost != null) {
        mPost.releaseConnection();
    }
}
}
```

learnFile

This web service will learn all the files present in the search classification and image classification learning folders of the batch class. This web service supports using a PDF file's EText layer for OCR, provided that the batch class has been configured accordingly to support that processing method.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/batchClass/learnFile/
<BatchClassIdentifier>

Sample client code using Apache Commons HttpClient

```
private static void fileLearningService() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/batchClass/learnFile/
{BatchClassIdentifier}";
    PostMethod mPost = new PostMethod(url);
    try {
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            String responseBody = mPost.getResponseAsString();
            // Generating result as responseBody.
            System.out.println(statusCode + " *** " + responseBody);
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(mPost.getResponseAsString());
        }
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (mPost != null) {
            mPost.releaseConnection();
        }
    }
}
```

```
    }  
  }  
}
```

learnFuzzyDatabase

This web service performs database learning for the Fuzzy DB feature. The database table to be learned is based on the Document Fuzzy and Field Fuzzy configurations defined inside the document type of the batch class. This web service emulates clicking the LearnDB command in the Transact user interface.

Request Method

POST

Web Service URL

http://<hostname>:<port>/dcma/rest/batchClass/learnFuzzyDatabase

Sample client code using Apache Commons HttpClient

```
private static void learnFuzzyDatabase() {  
    HttpClient client = new HttpClient();  
    String url = "http://localhost:8080/dcma/rest/learnFuzzyDatabase";  
    PostMethod mPost = new PostMethod(url);  
    // adding file for sending  
    Part[] parts = new Part[4];  
    try {  
        parts[0] = new StringPart("batchClassIdentifier", "BC1");  
        parts[1] = new StringPart("documentType", "Invoice-Table");  
        parts[2] = new StringPart("groupNames", "");  
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,  
mPost.getParams());  
        mPost.setRequestEntity(entity);  
        int statusCode = client.executeMethod(mPost);  
        if (statusCode == 200) {  
            System.out.println("Web service executed successfully.");  
            String responseBody = mPost.getResponseBodyAsString();  
            // Generating result as responseBody.  
            System.out.println(statusCode + " *** " + responseBody);  
        } else if (statusCode == 403) {  
            System.out.println("Invalid username/password.");  
        } else {  
            System.out.println(mPost.getResponseBodyAsString());  
        }  
    } catch (FileNotFoundException e) {  
        System.err.println("File not found for processing.");  
    } catch (HttpException e) {  
        e.printStackTrace();  
    } catch (IOException e) {  
        e.printStackTrace();  
    } finally {  
        if (mPost != null) {  
            mPost.releaseConnection();  
        }  
    }  
}
```

uploadLearningFile

This web service uploads learning files for one or more document types in a batch class. As opposed to the uploadLearnFile web service, which automatically places pages in the appropriate first/middle/last folders based on their positions in the uploaded files, this web service allows the developer to choose the exact folder (first, middle or last) where individual pages will be placed. The files are uploaded to one or more of the following folders depending on the 'learning_type' value defined in the web service parameters file.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/batchClass/uploadLearningFile

Sample client code using Apache Commons HttpClient

```
private static void learnFile() {
    HttpClient client = newHttpClient();
    String url = "http://localhost:8080/dcma/rest/batchClass/uploadLearningFile";
    PostMethod mPost = new PostMethod(url);
    // Adding Input XML file for processing
    File file = new File("C:\\sample\\Input.xml");
    File file1 = new File("C:\\sample\\first.tif");
    File file2 = new File("C:\\sample\\second.tif");
    Part[] parts = new Part[number of files need to be uploaded];
    try {
        parts[0] = new FilePart(file.getName(), file);
        parts[1] = new FilePart(file1.getName(), file1);
        parts[2] = new FilePart(file2.getName(), file2);
        parts[n] = new FilePart(filen.getName(), filen);
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        // send post request to server
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            String responseBody = mPost.getResponseBodyAsString();
            // Generating result as responseBody.
            System.out.println(statusCode + " *** " + responseBody);
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(mPost.getResponseBodyAsString());
        }
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (mPost != null) {
            mPost.releaseConnection();
        }
    }
}
```

uploadLearnFiles

This web service uploads files to the specified learning folder of a batch class based on the specified learning type. This web service approximates the process of learning files through the Transact user interface, where the user only needs to supply the batch class, document type, learning type, and files, and the web services will distribute the individual pages into the appropriate first, middle and last page folders in the operating system automatically.

Request Method

POST

Web Service URL`http://<serverName>:<port>/dcma/rest/batchClass/uploadLearnFiles`

Batch instance management web services

AddUserRolesToBatchInstance

This web service adds a user role to a batch instance.

Request Method

GET

Web Service URL`http://<serverName>:<port>/dcma/rest/AddUserRolesToBatchInstance/
<batchInstanceIdentifier>/<userRole>`**Sample client code using Apache Commons HttpClient**

```
private static void addUserRolesToBatchInstance() {
    HttpClient client = new HttpClient();
    // URL path to be hit for adding user roles to batch instance identifier
    String url = "http://localhost:8080/dcma/rest/addUserRolesToBatchInstance/BI45/
admin";
    GetMethod getMethod = new GetMethod(url);
    int statusCode;
    try {
        statusCode = client.executeMethod(getMethod);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            String responseBody = getMethod.getResponseBodyAsString();
            System.out.println(statusCode + " *** " + responseBody);
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(getMethod.getResponseBodyAsString());
        }
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (getMethod != null) {
            getMethod.releaseConnection();
        }
    }
}
```

```
}  
}
```

BatchInstanceList

This web service returns a list of batch instances with the specified status. If the user is the super admin user, all batch instances with the specified status will be returned. If the user is not the super admin user, only batch instances that can be accessed by the user will be returned. The following details will be returned for each batch instance: batch name, executed modules, local folder, review operator, batch identifier, drop folder.

Request Method

GET

Web Service URL

`http://<serverName>:<port>/dcma/rest/BatchInstanceList/<status>`

Sample client code using Apache Commons HttpClient

```
private static void getBatchInstanceList() {  
    HttpClient client = new HttpClient();  
    String url = "http://localhost:8080/dcma/rest/BatchInstanceList/RUNNING";  
    GetMethod getMethod = new GetMethod(url);  
    int statusCode;  
    try {  
        statusCode = client.executeMethod(getMethod);  
        if (statusCode == 200) {  
            System.out.println("Web service executed successfully.");  
            String responseBody = getMethod.getResponseBodyAsString();  
            System.out.println(statusCode + " *** " + responseBody);  
        } else if (statusCode == 403) {  
            System.out.println("Invalid username/password.");  
        } else {  
            System.out.println(getMethod.getResponseBodyAsString());  
        }  
    } catch (HttpException e) {  
        e.printStackTrace();  
    } catch (IOException e) {  
        e.printStackTrace();  
    } finally {  
        if (getMethod != null) {  
            getMethod.releaseConnection();  
        }  
    }  
}
```

deleteBatchInstance

This web service is used to delete a batch instance. The batch instance will be deleted only if the batch instance is not locked by any user, and is in one of the following states: Error, Running, Ready for Review, or Ready for Validation. Note that the user must be authenticated and must have access to the batch instance being deleted.

Request Method

GET

Web Service URL

http://<serverName>:<port>/dcma/rest/deleteBatchInstance/<identifier>

Sample client code using Apache Commons HttpClient

```
private static void deleteBatchInstance() {
    HttpClient client = new HttpClient();
    // User can only delete batch instances with statuses of ERROR, READY_FOR_REVIEW,
    READY_FOR_VALIDATION, or RUNNING
    String url = "http://localhost:8080/dcma/rest/deleteBatchInstance/BI1";
    GetMethod getMethod = new GetMethod(url);
    int statusCode;
    try {
        statusCode = client.executeMethod(getMethod);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            String responseBody = getMethod.getResponseBodyAsString();
            System.out.println(statusCode + " *** " + responseBody);
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(getMethod.getResponseBodyAsString());
        }
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (getMethod != null) {
            getMethod.releaseConnection();
        }
    }
}
```

getBatchInstanceForRole

This web service retrieves the list of batch instances accessible to a particular role.

Request Method

GET

Web Service URL

http://<serverName>:<port>/dcma/rest/getBatchInstanceForRole/<role>

Sample client code using Apache Commons HttpClient

```
private static void getBatchInstanceForRole() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/getBatchInstanceForRoles/admin";
    GetMethod getMethod = new GetMethod(url);
    int statusCode;
    try {
        statusCode = client.executeMethod(getMethod);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            String responseBody = getMethod.getResponseBodyAsString();
            System.out.println(statusCode + " *** " + responseBody);
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(getMethod.getResponseBodyAsString());
        }
    }
}
```



```
    }  
  } catch (HttpException e) {  
    e.printStackTrace();  
  } catch (IOException e) {  
    e.printStackTrace();  
  } finally {  
    if (getMethod != null) {  
      getMethod.releaseConnection();  
    }  
  }  
}
```

Batch instance processing web services

advancedBarcodeExtraction

This web service performs barcode extraction on the input document based on the configuration options provided in the parameters.xml file. The settings for the ADVANCED_BARCODE_EXTRACTION plugin for the supplied batch class identifier are retrieved and used to perform the extraction.

Request Method

POST

Web Service URL

http://<serverName>:8080/dcma/rest/advancedBarcodeExtraction

Sample client code using Apache Commons HttpClient

```
private static void advancedBarcodeExtraction() {  
    HttpClient client = new HttpClient();  
    String url = "http://localhost:8080/dcma/rest/advancedBarcodeExtraction";  
    PostMethod mPost = new PostMethod(url);  
    File file1 = new File("C:\\sample\\sample.xml");  
    // adding xml file for taking input  
    File file2 = new File("C:\\sample\\US-Invoice.tif");  
    Part[] parts = new Part[2];  
    try {  
        parts[0] = new FilePart(file1.getName(), file1);  
        parts[1] = new FilePart(file2.getName(), file2);  
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,  
mPost.getParams());  
        mPost.setRequestEntity(entity);  
        int statusCode = client.executeMethod(mPost);  
        if (statusCode == 200) {  
            System.out.println("Web service executed successfully.");  
            System.out.println(mPost.getResponseBodyAsString());  
        } else if (statusCode == 403) {  
            System.out.println("Invalid username/password.");  
        } else {  
            System.out.println(mPost.getResponseBodyAsString());  
        }  
    } catch (HttpException e) {  
        e.printStackTrace();  
    } catch (IOException e) {  
        e.printStackTrace();  
    } finally {  
        if (mPost != null) {  
            mPost.releaseConnection();  
        }  
    }  
}
```

```

    }
}

```

advancedUploadBatch

This web service uploads a new batch to the identified batch class's drop folder.

Request Method

POST

Request URL

<ProtocolType>://<serverName>:<ServerPort>/dcma/rest/advancedUploadBatch

Sample URL

http://localhost:8080/dcma/rest/advancedUploadBatch

Sample input XML

```

<Upload_Batch>
  <BatchClassIdentifier>BC1</BatchClassIdentifier>
  <BatchClassFields>
    <BatchClassField>
      <Name>Test1</Name>
      <Value>123</Value>
    </BatchClassField>
    <BatchClassField>
      <Name>Test2</Name>
      <Value>456</Value>
    </BatchClassField>
  </BatchClassFields>
  <BatchInstanceName>SampleBatch</BatchInstanceName>
  <BatchDescription>SampleDesc</BatchDescription>
  <BatchPriority>20</BatchPriority>
</Upload_Batch>

```

Sample client code using Apache Commons HttpClient

```

HttpClient client = new HttpClient();
String url = "http://localhost:8080/dcma/rest/advancedUploadBatch";
PostMethod mPost = new PostMethod(url);
File file1 = new File("C:\\sample\\advancedUploadBatch.xml");
File file2 = new File("C:\\sample\\US-Invoice.tif");
Part[] parts = new Part[2];
try {
    // Input XML
    parts[0] = new FilePart(file1.getName(), file1);
    // File for processing
    parts[1] = new FilePart(file2.getName(), file2);
    MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
    mPost.setRequestEntity(entity);
    int statusCode = client.executeMethod(mPost);
    if (statusCode == 200) {
        System.out.println("Web service executed successfully.");
        String responseBody = mPost.getResponseBodyAsString();
        System.out.println(statusCode + " *** " + responseBody);
    } else if (statusCode == 403) {
        System.out.println("Invalid username/password.");
    } else {
        System.out.println(mPost.getResponseBodyAsString());
    }
}

```

```
} catch (FileNotFoundException e) {
    System.err.println("File not found for processing.");
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (mPost != null) {
        mPost.releaseConnection();
    }
}
```

barcodeExtraction

This web service performs barcode extraction on the input document based on the configuration options provided in the parameters.xml file. The settings for the ADVANCED_BARCODE_EXTRACTION plugin for the supplied batch class identifier are retrieved and used to perform the extraction.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/barcodeExtraction

Sample client code using Apache Commons HttpClient

```
private static void barcodeExtraction() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/barcodeExtraction";
    PostMethod mPost = new PostMethod(url);
    File file1 = new File("C:\\sample\\sample.tif");
    // adding xml file for taking input
    File file2 = new File("C:\\sample\\WebServiceParams-barcodeExtraction.xml");
    Part[] parts = new Part[2];
    try {
        parts[0] = new FilePart(file1.getName(), file1);
        parts[1] = new FilePart(file2.getName(), file2);
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            System.out.println(mPost.getResponseBodyAsString());
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(mPost.getResponseBodyAsString());
        }
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (mPost != null) {
            mPost.releaseConnection();
        }
    }
}
```

checkWSStatus

This web service queries the current status of the initiateOcrClassifyExtract web service.

Request Method

GET

Web Service URL

http://<serverName>:<port>/dcma/rest/checkWSStatus

Sample client code using Apache Commons HttpClient

```
private static void checkWSStatus() {
    HttpClient client = new HttpClient();
    String url = " http://localhost:8080/dcma/rest/checkWSStatus?ocrToken=1233232323 ";
    GetMethod getMethod = new GetMethod(url);
    int statusCode;
    try {
        statusCode = client.executeMethod(getMethod);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            String responseBody = getMethod.getResponseBodyAsString();
            System.out.println(statusCode + " *** " + responseBody);
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(getMethod.getResponseBodyAsString());
        }
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (getMethod != null) {
            getMethod.releaseConnection();
        }
    }
}
```

classifyBarcodeImage

This web service is used to classify the supplied image using a bar code according to the specified batch class. The image file should contain a bar code and the bar code value should represent a document type that exists in the batch class.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/classifyBarcodeImage

Sample client code using Apache Commons HttpClient

```
private static void classifyBarcodeImage() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/classifyBarcodeImage";
    PostMethod mPost = new PostMethod(url);
}
```

```

File file1 = new File("C:\sample\US-Invoice.tif");
Part[] parts = new Part[2];
try {
    parts[0] = new FilePart(file1.getName(), file1);
    parts[1] = new StringPart("batchClassId", "BC1");
    MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
    mPost.setRequestEntity(entity);
    int statusCode = client.executeMethod(mPost);
    if (statusCode == 200) {
        System.out.println("Web service executed successfully.");
        String responseBody = mPost.getResponseBodyAsString();
        System.out.println(statusCode + " *** " + responseBody);
    } else if (statusCode == 403) {
        System.out.println("Invalid username/password.");
    } else {
        System.out.println(mPost.getResponseBodyAsString());
    }
} catch (FileNotFoundException e) {
    System.err.println("File not found for processing.");
} catch (HttpException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (mPost != null) {
        mPost.releaseConnection();
    }
}
}

```

classifyHocr

This web service classifies the input HOCR.xml according to the batch class identifier provided. The appropriate classification and assembly plugins must be configured in the batch class. Document learning must be performed on the batch class before the web service is used. If the batch class does not have the required plugins configured, the web service will not work.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/classifyHocr

Sample client code using Apache Commons HttpClient

```

private static void classifyHocr() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/classifyHocr";
    PostMethod mPost = new PostMethod(url);
    File file1 = new File("C:\sample\US-Invoice.html");
    Part[] parts = new Part[2];
    try {
        parts[0] = new FilePart(file1.getName(), file1);
        parts[1] = new StringPart("batchClassId", "BC1");
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {

```

```
        System.out.println("Web service executed successfully..");
        String responseBody = mPost.getResponseBodyAsString();
        System.out.println(statusCode + " *** " + responseBody);
    } else if (statusCode == 403) {
        System.out.println("Invalid username/password.");
    } else {
        System.out.println(mPost.getResponseBodyAsString());
    }
} catch (FileNotFoundException e) {
    System.err.println("File not found for processing.");
} catch (HttpException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (mPost != null) {
        mPost.releaseConnection();
    }
}
}
```

classifyImage

This web service allows developers to classify documents from third party applications without creating batch instances. It performs layout-based classification based on the samples provided in the batch class and returns the document type. The input file to be classified must be a single-page TIFF file. This web service depends on the CREATE_THUMBNAILS, CLASSIFY_IMAGES and DOCUMENT_ASSEMBLER plugins. If the batch class does not contain these plugins the web service will not work.

Request Method

POST

Web Service URL

<http://<serverName>:<port>/dcma/rest/classifyImage>

Sample client code using Apache Commons HttpClient

```
private static void classifyImage() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/classifyImage";
    PostMethod mPost = new PostMethod(url);
    // Adding tif file for processing
    File file1 = new File("C:\\sample\\US-Invoice.tif");
    Part[] parts = new Part[2];
    try {
        parts[0] = new FilePart(file1.getName(), file1);
        // Adding parameter for batchClassId
        parts[1] = new StringPart("batchClassId", "BC1");
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully..");
            String responseBody = mPost.getResponseBodyAsString();
            System.out.println(statusCode + " *** " + responseBody);
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password..");
        } else {

```

```
        System.out.println(mPost.getResponseBodyAsString());
    }
} catch (FileNotFoundException e) {
    System.err.println("File not found for processing..");
} catch (HttpException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (mPost != null) {
        mPost.releaseConnection();
    }
}
}
```

classifyMultiPageHocr

This web service classifies multiple HOCR.xml files using the batch class identifier provided. The batch class depends on the SEARCH_CLASSIFICATION and DOCUMENT_ASSEMBLER plugins. Document learning must have been performed previously in the batch class.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/classifyMultiPageHocr

Sample client code using Apache Commons HttpClient

```
private static void classifyMultiPageHocr() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/classifyMultiPageHocr";
    PostMethod mPost = new PostMethod(url);
    File file1 = new File("D:\\sample\\New folder.zip");
    Part[] parts = new Part[2];
    try {
        parts[0] = new FilePart(file1.getName(), file1);
        parts[1] = new StringPart("batchClassId", "BC1");
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        String responseBody = mPost.getResponseBodyAsString();
        System.out.println(statusCode + "****" + responseBody);
        mPost.releaseConnection();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

createHOCRforBatchClass

This web service generates HOCR.xml files for TIFF or PDF files.

Request Method

POST

Web Service URL`http://<serverName>:<port>/dcma/rest/batchClass/createHOCRforBatchClass`**Sample client code using Apache Commons HttpClient**

```
private static void createHOCRXML() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/batchClass/createHOCRforBatchClass";
    PostMethod mPost = new PostMethod(url);
    // Adding Input XML file for processing
    File file = new File("C:\\sample\\Input.xml");
    File imageFile = new File("C:\\sample\\SampleImage.tif");
    Part[] parts = new Part[2];
    try {
        parts[0] = new FilePart(file.getName(), file);
        parts[1] = new FilePart(imageFile.getName(), imageFile);
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            String responseBody = mPost.getResponseBodyAsString();
            // Generating result as responseBody.
            System.out.println(statusCode + " *** " + responseBody);
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(mPost.getResponseBodyAsString());
        }
    } catch (FileNotFoundException e) {
        System.err.println("File not found for processing.");
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (mPost != null) {
            mPost.releaseConnection();
        }
    }
}
```

createOCR

This web service will create an HOCR.xml file for the provided image file. Tesseract, OmniPage and RecoStar all support conventional OCR processes where a non-TIFF image is converted to TIFF and then OCR'd accordingly. When RecoStar is used as the OCR engine, the EText layer of a PDF file can be used for OCR purposes if the RSP file has ExtractFromEText chosen as the processing mode. If a PNG or TIFF file is processed by the RecoStar OCR engine using an ExtractFromEText RSP file, a blank HOCR.xml file will be returned.

This web service supports using a PDF file's EText layer for OCR and extraction, provided that the batch class has been configured accordingly to support that processing method.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/createOCR

Sample client code using Apache Commons HttpClient

```
private static void createOCR() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/createOCR";
    PostMethod mPost = new PostMethod(url);
    File file1 = new File("C:\\sample\\sample1.tif");
    File file2 = new File("C:\\sample\\WebServiceParams.xml");
    File file3 = new File("C:\\sample\\Fpr.rsp");
    Part[] parts = new Part[3];
    try {
        parts[0] = new FilePart(file1.getName(), file1);
        parts[1] = new FilePart(file2.getName(), file2);
        parts[2] = new FilePart(file3.getName(), file3);
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            InputStream in = mPost.getResponseBodyAsStream();
            // saving result generated.
            File outputFile = new File("C:\\sample\\serverOutput.zip");
            FileOutputStream fos = new FileOutputStream(outputFile);
            try {
                byte[] buf = new byte[1024];
                int len = in.read(buf);
                while (len > 0) {
                    fos.write(buf, 0, len);
                    len = in.read(buf);
                }
            } finally {
                if (fos != null) {
                    fos.close();
                }
            }
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(mPost.getResponseBodyAsString());
        }
    } catch (FileNotFoundException e) {
        System.err.println("File not found for processing.");
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (mPost != null) {
            mPost.releaseConnection();
        }
    }
}
```

decryptBatchInstanceHocrXml

This web service decrypts an HOCR.xml file in a batch instance. When a batch is executed in an encrypted batch class, the HOCR.xml files created for the image files are also encrypted. The name of the HOCR.xml file and the batch instance identifier are passed in as parameters.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/decryptBatchInstanceHocrXml

Sample client code using Apache Commons HttpClient

```
private static void decryptBatchInstanceHocrXml() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/decryptBatchInstanceHocrXml";
    PostMethod mPost = new PostMethod(url);
    mPost.setDoAuthentication(true);
    Part[] parts = new Part[2];
    try {
        parts[0] = new StringPart("hocrFileName", "BI3_0_HOCR.xml");
        parts[1] = new StringPart("batchInstanceIdentifier", "BI3");
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            System.out.println("HOCR XML decrypted successfully");
            System.out.println(mPost.getResponseBodyAsString());
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(mPost.getResponseBodyAsString());
        }
    } catch (FileNotFoundException e) {
        System.out.println("File not found for processing.");
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (mPost != null) {
            mPost.releaseConnection();
        }
    }
}
```

decryptBatchXml

This web service decrypts an encrypted batch.xml file (generally found in the final drop folder of a batch instance). Encrypted batch.xml files can only be decrypted on Transact servers that use the same encryption keys as the server that originally encrypted the batch.xml file.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/decryptBatchXml

Sample client code using Apache Commons HttpClient

```
private static void decryptBatchXml() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/decryptBatchXml";
    PostMethod mPost = new PostMethod(url);
    mPost.setDoAuthentication(true);
    File file1 = new File("F:\\Ephesoft\\SharedFolders\\BC5\\Final-drop-folder
\\BI2\\BI2_batch.xml");
    Part[] parts = new Part[1];
    try {
        parts[0] = new FilePart(file1.getName(), file1);
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            System.out.println("Batch XML decrypted successfully");
            System.out.println(mPost.getResponseBodyAsString());
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(mPost.getResponseBodyAsString());
        }
    } catch (FileNotFoundException e) {
        System.out.println("File not found for processing.");
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (mPost != null) {
            mPost.releaseConnection();
        }
    }
}
```

decryptLuceneClassificationHocrXml

This web service decrypts an HOCR.xml file in the lucene-search-classification-sample folder within the batch class folder in the SharedFolders area.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/decryptLuceneClassificationHocrXml

Sample client code using Apache Commons HttpClient

```
private static void decryptLuceneClassificationHocrXml() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/decryptLuceneClassificationHocrXml";
    PostMethod mPost = new PostMethod(url);
    mPost.setDoAuthentication(true);
    // Input zip file for importing batch class.
    Part[] parts = new Part[4];
}
```

```

try {
    parts[0] = new StringPart("hocrFileName", "US-Invoice_HOCR.xml");
    parts[1] = new StringPart("batchClassIdentifier", "BC5");
    parts[2] = new StringPart("pageType", "First");
    parts[3] = new StringPart("documentType", "US Invoice");
    MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
    mPost.setRequestEntity(entity);
    int statusCode = client.executeMethod(mPost);
    if (statusCode == 200) {
        System.out.println("HOCR XML decrypted successfully");
        String responseBody = mPost.getResponseBodyAsString();
        System.out.println(statusCode + " *** " + responseBody);
    } else if (statusCode == 403) {
        System.out.println("Invalid username/password.");
    } else {
        System.out.println(mPost.getResponseBodyAsString());
    }
} catch (FileNotFoundException e) {
    System.out.println("File not found for processing.");
} catch (HttpException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (mPost != null) {
        mPost.releaseConnection();
    }
}
}

```

decryptTestHocrXml

This web service decrypts an HOCR.xml file in one of the test folders within the batch class. Test folders that can be used with the web service are test-advanced-extraction, test-classification, test-extraction and test-table.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/decryptTestHocrXml

Sample client code using Apache Commons HttpClient

```

private static void decryptTestHocrXml() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/decryptTestHocrXml";
    PostMethod mPost = new PostMethod(url);
    mPost.setDoAuthentication(true);
    Part[] parts = new Part[4];
    try {
        parts[0] = new StringPart("hocrFileName", "US-Invoice-0000_HOCR.xml");
        parts[1] = new StringPart("batchClassIdentifier", "BC5");
        parts[2] = new StringPart("testType", "table");
        parts[3] = new StringPart("documentType", "US Invoice");
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
    }
}

```

```

        if (statusCode == 200) {
            System.out.println("HOCR XML decrypted successfully");
            String responseBody = mPost.getResponseBodyAsString();
            System.out.println(statusCode + " *** " + responseBody);
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(mPost.getResponseBodyAsString());
        }
    } catch (FileNotFoundException e) {
        System.out.println("File not found for processing.");
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (mPost != null) {
            mPost.releaseConnection();
        }
    }
}

```

extractFieldFromHocr

This web service executes key-value extraction for the requested field based on key-value rules defined in the batch class.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/extractFieldFromHocr

Sample client code using Apache Commons HttpClient

```

private static void extractFieldFromHocr() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/extractFieldFromHocr";
    PostMethod mPost = new PostMethod(url);
    File file1 = new File("C:\\sample\\Application-Checklist-hocr.xml");
    Part[] parts = new Part[2];
    try {
        parts[0] = new FilePart(file1.getName(), file1);
        // Adding field value for extracting Key Value Pattern.
        parts[1] = new StringPart("fieldValue", "APPLICATION");
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            String responseBody = mPost.getResponseBodyAsString();
            System.out.println(statusCode + " *** " + responseBody);
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(mPost.getResponseBodyAsString());
        }
    } catch (FileNotFoundException e) {
        System.err.println("File not found for processing.");
    } catch (HttpException e) {

```

```
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (mPost != null) {
            mPost.releaseConnection();
        }
    }
}
```

extractFields

This web service provides multiple ways to perform extraction. The header parameter `extractionAPI` specifies the type of extraction to use.

Request Method

POST

Web Service URL

`http://<serverName>:<port>/dcma/rest/extractFields`

Sample client code using Apache Commons HttpClient using the `REGULAR_REGEX_EXTRACTION` option

```
private static void extractFields() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/extractFields";
    PostMethod mPost = new PostMethod(url);
    File file1 = new File("C:\\sample\\input\\sample1.html");
    // adding xml file for taking input
    File file2 = new File("C:\\sample\\input\\WebServiceParams.xml");
    Part[] parts = new Part[2];
    try {
        parts[0] = new FilePart(file1.getName(), file1);
        parts[1] = new FilePart(file2.getName(), file2);
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        // Pass the name of extraction api that is to be used: BARCODE_EXTRACTION,
RECCOSTAR_EXTRACTION, REGULAR_REGEX_EXTRACTION, KV_EXTRACTION, FUZZY_DB,
OMNIPAGE_EXTRACTION
        Header header = new Header("extractionAPI", "REGULAR_REGEX_EXTRACTION");
        mPost.setRequestHeader(header);
        mPost.setRequestEntity(entity);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            System.out.println(mPost.getResponseBodyAsString());
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(mPost.getResponseBodyAsString());
        }
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (mPost != null) {
            mPost.releaseConnection();
        }
    }
}
```

```
}
```

extractFieldsUsingRegex

This web service extracts values and coordinates based on regular expressions defined for all fields of a particular document type of a particular batch class.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/extractFieldsUsingRegex

Sample client code using Apache Commons HttpClient

```
private static void extractFieldsUsingRegex() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/extractFieldsUsingRegex";
    PostMethod mPost = new PostMethod(url);
    File file1 = new File("C:\\sample\\sample1.xml");
    File file2 = new File("C:\\sample\\WebServiceParams.xml");
    Part[] parts = new Part[3];
    try {
        parts[0] = new FilePart(file1.getName(), file1);
        parts[1] = new FilePart(file2.getName(), file2);
        parts[2] = new StringPart("hocrFileName", file1.getName());
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            System.out.println(mPost.getResponseBodyAsString());
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(mPost.getResponseBodyAsString());
        }
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (mPost != null) {
            mPost.releaseConnection();
        }
    }
}
```

extractFixedForm

This web service provides two different processing options for extracting fixed-form data. Different XML parameter input file formats are needed depending on the option chosen.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/extractFixedForm

Sample client code using Apache Commons HttpClient

```
private static void extractFixedForm() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/extractFixedForm";
    PostMethod mPost = new PostMethod(url);
    // adding file for sending
    File file1 = new File("C:\\sample\\fixedForm.xml");
    File file2 = new File("C:\\sample\\Image.tif");
    Part[] parts = new Part[2];
    try {
        parts[0] = new FilePart(file1.getName(), file1);
        parts[1] = new FilePart(file2.getName(), file2);
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            String responseBody = mPost.getResponseBodyAsString();
            // Generating result as responseBody.
            System.out.println(statusCode + " *** " + responseBody);
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(mPost.getResponseBodyAsString());
        }
    } catch (FileNotFoundException e) {
        System.err.println("File not found for processing.");
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (mPost != null) {
            mPost.releaseConnection();
        }
    }
}
```

extractFuzzyDB

This web service extracts index field information using the Fuzzy Database Lookup feature (based on the Document Fuzzy configurations).

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/extractFuzzyDB

Sample client code using Apache Commons HttpClient

```
private static void extractFuzzyDB() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/extractFuzzyDB";
    PostMethod mPost = new PostMethod(url);
```



```

File file = new File("C:\sample\Application-Checklist_000-hocr.xml");
Part[] parts = new Part[4];
try {
    parts[0] = new FilePart(file.getName(), file);
    parts[1] = new StringPart("documentType", "Application-Checklist");
    parts[2] = new StringPart("batchClassIdentifier", "BC1");
    parts[3] = new StringPart("hocrFile", "Application-Checklist_000-hocr.xml");
    MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
mPost.setRequestEntity(entity);
int statusCode = client.executeMethod(mPost);
if (statusCode == 200) {
    System.out.println("Web service executed successfully.");
    String responseBody = mPost.getResponseBodyAsString();
    // Generating result as responseBody.
    System.out.println(statusCode + " *** " + responseBody);
} else if (statusCode == 403) {
    System.out.println("Invalid username/password.");
} else {
    System.out.println(mPost.getResponseBodyAsString());
}
} catch (FileNotFoundException e) {
    System.err.println("File not found for processing.");
} catch (HttpException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (mPost != null) {
        mPost.releaseConnection();
    }
}
}

```

extractFieldsForFuzzyDB

This web service extracts index field values utilizing both document- and field-level fuzzy database configurations.

Request Method

POST

Web Service URL

http://<serverName>:8080/dcma/rest/extractFieldsForFuzzyDB

Sample client code using Apache Commons HttpClient

```

private static void extractFieldsForFuzzyDB() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/extractFieldsForFuzzyDB";
    PostMethod mPost = new PostMethod(url);
    File webServInputFile = new File("C:\sample\FuzzyDBExtract_Input.xml");
    File hocrZipFile = new File("C:\sample\HOCR_XML.zip");
    Part[] parts = new Part[2];
    try {
        parts[0] = new FilePart(webServInputFile.getName(), webServInputFile);
        parts[1] = new FilePart(hocrZipFile.getName(), hocrZipFile);
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
mPost.setRequestEntity(entity);
int statusCode = client.executeMethod(mPost);

```

```
    if (statusCode == 200) {
        System.out.println("Web service executed successfully.");
        String responseBody = mPost.getResponseBodyAsString();
        // Generating result as responseBody.
        System.out.println(statusCode + " *** " + responseBody);
    } else if (statusCode == 403) {
        System.out.println("Invalid username/password.");
    } else {
        System.out.println(mPost.getResponseBodyAsString());
    }
} catch (FileNotFoundException e) {
    System.err.println("File not found for processing.");
} catch (HttpException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (mPost != null) {
        mPost.releaseConnection();
    }
}
}
```

extractKV

This web service extracts the document-level fields using key-value extraction rule properties specified in the parameters.xml file, using an HOcr.xml file as input. If the key-value pattern does not find a match in the HOcr.xml file, empty document-level fields will be created. For best results we recommend that you test the configuration options in the Transact user interface before building the configuration files for use with the web service.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/extractKV

Sample client code using Apache Commons HttpClient

```
private static void extractKV() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/extractKV";
    PostMethod mPost = new PostMethod(url);
    File f1 = new File("C:\\sample\\extractKV.xml");
    File f2 = new File("C:\\sample\\Application-Checklist-hocr.xml");
    Part[] parts = new Part[3];
    try {
        parts[0] = new FilePart(f1.getName(), f1);
        parts[1] = new FilePart(f2.getName(), f2);
        parts[2] = new StringPart("hocrFileName", f2.getName());
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            String responseBody = mPost.getResponseBodyAsString();
            // Generating result as responseBody.
            System.out.println(statusCode + " *** " + responseBody);
        } else if (statusCode == 403) {
```

```
        System.out.println("Invalid username/password.");
    } else {
        System.out.println(mPost.getResponseBodyAsString());
    }
} catch (FileNotFoundException e) {
    System.err.println("File not found for processing.");
} catch (HttpException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (mPost != null) {
        mPost.releaseConnection();
    }
}
}
```

ExtractKVForDocumentType

This web service performs key-value extraction on an HOCR.xml file. The batch class ID and document type identified in the parameters.xml file are used to identify the key-value extraction rules that will be executed.

Request Method

POST

Web Service URL

`http://<serverName>:<port>/dcma/rest/batchClass/ExtractKVForDocumentType`

Sample client code using Apache Commons HttpClient

```
private static void performKeyValueExtraction() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/batchClass/ExtractKVForDocumentType";
    PostMethod mPost = new PostMethod(url);
    File file = new File("C:\\sample\\Input.xml");
    File hocrZipFile = new File("C:\\sample\\Input_HOCR.zip");
    Part[] parts = new Part[2];
    try {
        parts[0] = new FilePart(file.getName(), file);
        parts[1] = new FilePart(hocrZipFile.getName(), hocrZipFile);
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            String responseBody = mPost.getResponseBodyAsString();
            // Generating result as responseBody.
            System.out.println(statusCode + " *** " + responseBody);
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(mPost.getResponseBodyAsString());
        }
    } catch (FileNotFoundException e) {
        System.err.println("File not found for processing.");
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
```

```
        e.printStackTrace();
    } finally {
        if (mPost != null) {
            mPost.releaseConnection();
        }
    }
}
```

initiateOcrClassifyExtract

This web service performs extraction on the supplied documents. The extraction plugins are retrieved from the batch class corresponding to the supplied batch class identifier, and classification and extraction will be performed based on those plugins. If the document type is provided as a parameter, classification will be skipped, and only extraction will be performed.

Request Method

POST

Web Service URL

`http://<serverName>:<port>/dcma/rest/initiateOcrClassifyExtract`

Checklist

- Extraction would be done only if Extraction module is configured for the particular batch class
- Extraction would be performed only for the plugins which have extraction switch ON in batch class configuration.

Sample client code using Apache Commons HttpClient

```
private static void initiateOcrClassifyExtract() {
    HttpClient client = new HttpClient();
    Credentials defaultcreds = new UsernamePasswordCredentials("username", "password");
    client.getState().setCredentials(new AuthScope("serverName", 8080), defaultcreds);
    client.getParams().setAuthenticationPreemptive(true);
    String url = "http://localhost:8080/dcma/rest/ initiateOcrClassifyExtract";
    PostMethod mPost = new PostMethod(url);
    // Adding HTML file for processing
    File file1 = new File("C:\\sample\\US-Invoice.tiff");
    Part[] parts = new Part[2];
    try {
        parts[0] = new FilePart(file1.getName(), file1);
        // Adding parameter for batchClassIdentifier
        parts[1] = new StringPart("batchClassIdentifier", "BC1");
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully..");
            String responseBody = mPost.getResponseBodyAsString();
            System.out.println(statusCode + " *** " + responseBody);
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password..");
        } else {
            System.out.println(mPost.getResponseBodyAsString());
        }
    } catch (FileNotFoundException e) {
        System.err.println("File not found for processing..");
    } catch (HttpException e) {
        e.printStackTrace();
    }
}
```

```
} catch (IOException e) {  
    e.printStackTrace();  
} finally {  
    if (mPost != null) {  
        mPost.releaseConnection();  
    }  
}  
}
```

keywordClassification

This web service classifies images into documents based on key-value page process rules (both page-level fields and classification rules). If classification is successful an XML object containing all documents is sent back as a response. If the KV_PAGE_PROCESS plugin is not configured the web service will return an error code.

This web service supports using a PDF file's EText layer for OCR and extraction, provided that the batch class has been configured accordingly to support that processing method.

Request Method

POST

Web Service URL

`http://<serverName>:<port>/dcma/rest/keywordClassification`

ocrClassify

This web service performs OCR and classification on the provided documents.

Request Method

POST

Web Service URL

`http://<serverName>:<port>/dcma/rest/ocrClassify`

ocrClassifyExtract

This web service performs classification and extraction on the supplied documents. Classification and extraction are performed based on the plugin configurations of the batch class identifier supplied. This web service supports using a PDF file's EText layer for OCR and extraction, provided that the batch class has been configured accordingly to support that processing method.

Request Method

POST

Web Service URL

`http://<serverName>:<port>/dcma/rest/ocrClassifyExtract`

Sample client code using Apache Commons HttpClient

```
private static void ocrClassifyExtract() {  
    HttpClient client = new HttpClient();  
    String url = "http://localhost:8080/dcma/rest/ocrClassifyExtract";
```

```

PostMethod mPost = new PostMethod(url);
// Adding HTML file for processing
File file1 = new File("C:\sample\US-Invoice.tiff");
Part[] parts = new Part[2];
try {
    parts[0] = new FilePart(file1.getName(), file1);
    // Adding parameter for batchClassIdentifier
    parts[1] = new StringPart("batchClassIdentifier", "BC1");
    MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
    mPost.setRequestEntity(entity);
    int statusCode = client.executeMethod(mPost);
    if (statusCode == 200) {
        System.out.println("Web service executed successfully..");
        String responseBody = mPost.getResponseBodyAsString();
        System.out.println(statusCode + " *** " + responseBody);
    } else if (statusCode == 403) {
        System.out.println("Invalid username/password..");
    } else {
        System.out.println(mPost.getResponseBodyAsString());
    }
} catch (FileNotFoundException e) {
    System.err.println("File not found for processing..");
} catch (HttpException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (mPost != null) {
        mPost.releaseConnection();
    }
}
}

```

OcrClassifyExtractSearchablePDF

This web service performs OCR, classification and extraction on the input documents, creating searchable PDF files and a batch.xml file containing the extraction results. Classification and extraction will be performed based on the plugins configured in the batch class.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/OcrClassifyExtractSearchablePDF

Sample client code using Apache Commons HttpClient

```

private static void OcrClassifyExtractSearchablePDF() {
    HttpClient client = new HttpClient();
    Credentials defaultcreds = new UsernamePasswordCredentials("ephesoft", "demo");
    client.getState().setCredentials(new AuthScope("localhost", 8080), defaultcreds);
    client.getParams().setAuthenticationPreemptive(true);
    String url = "http://localhost:8080/dcma/rest/OcrClassifyExtractSearchablePDF";
    PostMethod mPost = new PostMethod(url);
    // Adding HTML file for processing
    File file1 = new File("C:\sample\US-Invoice.tif");
    Part[] parts = new Part[2];
    try {
        parts[0] = new FilePart(file1.getName(), file1);

```

```

// Adding parameter for batchClassIdentifier
parts[1] = new StringPart("batchClassIdentifier", "BC5");
MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
mPost.setRequestEntity(entity);
int statusCode = client.executeMethod(mPost);
if (statusCode == 200) {
    System.out.println("Batch class exported successfully");
    InputStream in = mPost.getResponseBodyAsStream();
    File f = new File("C:\\sample\\Output.zip");
    FileOutputStream fos = new FileOutputStream(f);
    try {
        byte[] buf = new byte[1024];
        int len = in .read(buf);
        while (len > 0) {
            fos.write(buf, 0, len);
            len = in .read(buf);
        }
    } finally {
        if (fos != null) {
            fos.close();
        }
    }
} else if (statusCode == 403) {
    System.out.println("Invalid username/password.");
} else {
    System.out.println(mPost.getResponseBodyAsString());
}
} catch (FileNotFoundException e) {
    System.out.println("File not found for processing..");
} catch (HttpException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (mPost != null) {
        mPost.releaseConnection();
    }
}
}

```

restartAllBatchInstance

This web service restarts all batch instances with states of Ready for Review or Ready for Validation. The RestartAll switch must be set to TRUE in the properties file for this web service to work.

Request Method

GET

Web Service URL

http://<serverName>:<port>/dcma/rest/restartAllBatchInstance

Sample client code using Apache Commons HttpClient

```

private static void restartAllBatchInstance() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/restartAllBatchInstance";
    GetMethod getMethod = new GetMethod(url);
    int statusCode;
    try {
        statusCode = client.executeMethod(getMethod);
    }
}

```

```
    if (statusCode == 200) {
        System.out.println("Web service executed successfully.");
        String responseBody = getMethod.getResponseBodyAsString();
        System.out.println(statusCode + " *** " + responseBody);
    } else if (statusCode == 403) {
        System.out.println("Invalid username/password.");
    } else {
        System.out.println(getMethod.getResponseBodyAsString());
    }
} catch (HttpException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (getMethod != null) {
        getMethod.releaseConnection();
    }
}
}
```

restartBatchInstance

This web service restarts a batch from the specified module. The user can only restart batch instances that are accessible to their role. Only batches in the Running, Error, Ready for Review, or Ready for Validation states can be restarted.

Request Method

GET

Web Service URL

http://<serverName>:<port>/dcma/rest/restartBatchInstance/
<batchInstanceIdentifier>/<restartAtModuleName>

Sample client code using Apache Commons HttpClient

```
private static void restartBatchInstance() {
    HttpClient client = new HttpClient();
    String url = "http://{serverName}:{port}/dcma/rest/restartBatchInstance/BI1/  
Folder_Import_Module";
    GetMethod getMethod = new GetMethod(url);
    int statusCode;
    try {
        statusCode = client.executeMethod(getMethod);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            String responseBody = getMethod.getResponseBodyAsString();
            System.out.println(statusCode + " *** " + responseBody);
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(getMethod.getResponseBodyAsString());
        }
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (getMethod != null) {
            getMethod.releaseConnection();
        }
    }
}
```



```
}  
}
```

runBatchInstance

This web service takes a batch instance in the Ready for Review or Ready for Validation state and moves it to the next phase in the workflow. The user can only run batch instances that are accessible to their role.

Request Method

GET

Web Service URL

`http://<serverName>:<port>/dcma/rest/runBatchInstance/
<batchInstanceIdentifier>`

Sample client code using Apache Commons HttpClient

```
private String runBatchInstance() {  
    String webserviceURL = "http://localhost:8080/dcma/rest/runBatchInstance/BI15";  
    HttpClient httpClient = new HttpClient();  
    GetMethod mget = new GetMethod(webserviceURL);  
    String webserviceResponse = "";  
    try {  
        int statusCode = httpClient.executeMethod(mget);  
        System.out.println("Webservice status code: " + statusCode);  
        if (statusCode == 200) {  
            System.out.println("Web service executed successfully.");  
            String responseBody = mget.getResponseBodyAsString();  
            // Generating result as responseBody.  
  
            System.out.println(mget.getResponseBodyAsString());  
            webserviceResponse = responseBody;  
  
            mget.releaseConnection();  
            return webserviceResponse;  
        }  
    } catch (HttpException e) {  
        e.printStackTrace();  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
    return webserviceResponse;  
}
```

searchTextForDocFuzzy

This web service searches fuzzy database index data to look for matches with the supplied search text. This web service emulates the document-level Fuzzy DB search field in the Validation screen in the Transact user interface.

Request Method

POST

Web Service URL

`http://<serverName>:<port>/dcma/rest/searchTextForDocFuzzy`

Sample client code using Apache Commons HttpClient

```
private static void searchTextForDocFuzzy() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/searchTextForDocFuzzy";
    PostMethod mPost = new PostMethod(url);
    Part[] parts = new Part[4];
    try {
        parts[0] = new StringPart("batchClassIdentifier", "BC1");
        parts[1] = new StringPart("documentType", "Invoice-Table");
        parts[2] = new StringPart("searchText", "*");
        parts[3] = new StringPart("searchType", "0");
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            String responseBody = mPost.getResponseBodyAsString();
            // Generating result as responseBody.
            System.out.println(statusCode + " *** " + responseBody);
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(mPost.getResponseBodyAsString());
        }
    } catch (FileNotFoundException e) {
        System.err.println("File not found for processing.");
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (mPost != null) {
            mPost.releaseConnection();
        }
    }
}
```

searchTextForFieldFuzzy

This web service searches fuzzy database index data to look for matches with the supplied index field-level text. This web service emulates the field-level Fuzzy DB search field in the Validation screen in the Transact user interface.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/searchTextForFieldFuzzy

Sample client code using Apache Commons HttpClient

```
private static void searchTextForDocFuzzy() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/searchTextForDocFuzzy";
    PostMethod mPost = new PostMethod(url);
    Part[] parts = new Part[4];
    try {
        parts[0] = new StringPart("batchClassIdentifier", "BC1");
        parts[1] = new StringPart("documentType", "Invoice-Table");
```

```

        parts[2] = new StringPart("searchText", "*");
        parts[3] = new StringPart("searchType", "0");
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            String responseBody = mPost.getResponseBodyAsString();
            // Generating result as responseBody.
            System.out.println(statusCode + " *** " + responseBody);
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(mPost.getResponseBodyAsString());
        }
    } catch (FileNotFoundException e) {
        System.err.println("File not found for processing.");
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (mPost != null) {
            mPost.releaseConnection();
        }
    }
}

```

setBatchClassField

This web service creates batch class fields in the supplied batch class identifier.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/setBatchClassField

Sample client code using Apache Commons HttpClient

```

private static void setBatchClassField() {
    HttpClient client = new HttpClient();
    Credentials defaultcreds = new UsernamePasswordCredentials("ephesoft",
"demo");
    client.getState().setCredentials(new AuthScope("localhost", 8080),
defaultcreds);
    client.getParams().setAuthenticationPreemptive(true);
    String url = "http://localhost:8080/dcma/rest/setBatchClassField";
    PostMethod mPost = new PostMethod(url);
    File file1 = new File("C:\\EpheWSTest\\setBatchClassField\\parameters.xml");
    Part[] parts = new Part[1];

    try {
        parts[0] = new FilePart(file1.getName(), file1);
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        System.out.println("Status code is " + statusCode);
    }
}

```

```
if (statusCode == 200) {
String responseBody = mPost.getResponseBodyAsString();
System.out.println(statusCode + " *** " + responseBody);
} else if (statusCode == 403) {
System.out.println("Invalid username/password..");
} else {
System.out.println(mPost.getResponseBodyAsString());
}

} catch (FileNotFoundException e) {
System.out.println("File not found for processing..");
} catch (HttpException e) {
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
} finally {

if (mPost != null) {
mPost.releaseConnection();
}
}
}
```

tableExtractionHOcr

This web service will extract table data from the HOcr.xml file.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/batchClass/tableExtractionHOcr

Sample client code using Apache Commons HttpClient

```
private static void tableExtractionHOcr() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/batchClass/tableExtractionHOcr";
    PostMethod mPost = new PostMethod(url);
    // adding file for sending
    File file1 = new File("C:\\sample\\sample.xml");
    File file2 = new File("C:\\sample\\US-Invoice_HOcr.xml");
    Part[] parts = new Part[2];
    try {
        parts[0] = new FilePart(file1.getName(), file1);
        parts[1] = new FilePart(file2.getName(), file2);
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            String responseBody = mPost.getResponseBodyAsString();
            // Generating result as responseBody.
            System.out.println(statusCode + " *** " + responseBody);
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(mPost.getResponseBodyAsString());
        }
    } catch (FileNotFoundException e) {
```

```
        System.err.println("File not found for processing.");
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (mPost != null) {
            mPost.releaseConnection();
        }
    }
}
```

uploadBatch

This web service uploads a batch for a given batch class. The uploaded file is copied to the batch class's drop folder as a single folder. The user must be authorized to execute a batch instance in that batch class otherwise an error message will be generated.

Request Method

POST

Web Service URL

`http://<serverName>:<port>/dcma/rest/uploadBatch/<batchClassIdentifier>/<batchInstanceName>`

Sample client code using Apache Commons HttpClient

```
private static void uploadBatch() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/uploadBatch/{BatchClassIdentifier}/
{BatchInstanceName} ";
    PostMethod mPost = new PostMethod(url);
    File file1 = new File("C:\\sample\\sample1.tif");
    Part[] parts = new Part[1];
    try {
        parts[0] = new FilePart(file1.getName(), file1);
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        String responseBody = mPost.getResponseBodyAsString();
        // Generating result as responseBody.
        System.out.println(statusCode + "****" + responseBody);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(mPost.getResponseBodyAsString());
        }
    } catch (FileNotFoundException e) {
        System.err.println("File not found for processing.");
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (mPost != null) {
            mPost.releaseConnection();
        }
    }
}
```

```
}  
}
```

v2/ocrClassifyExtract

This web service was designed primarily to work with the Nintex workflow engine. It performs OCR, classification and extraction on the supplied input documents. Classification is based only on the first page of each file before extracting the index fields defined in the corresponding batch class. Classification and extraction are performed based on the plugins configured in the corresponding batch class, with two exceptions: 1) The PAGE_PROCESS_SCRIPTING_PLUGIN plugin will not be executed; and 2) Table extraction will be performed if the plugin is turned on, but extracted table data will not be returned in the response.

Only one file can be supplied as input, but multiple files can be processed at one time by combining them in a zip file and submitting the zip file to the web service. However, note that each physical file inside the zip file will be treated as a single logical file, and will be classified based on the contents of the first page only (no separation will occur within the individual files).

This web service supports using a PDF file's EText layer for OCR and extraction, provided that the batch class has been configured accordingly to support that processing method.

Request Method

POST

Web Service URL

`http://<serverName>:<port>/dcma/rest/v2/ocrClassifyExtract`

v2/ocrClassifyExtractBase64

This web service performs the same functionality as the v2/ocrClassifyExtract web service, but it uses a Base64-encoded string as input. Many cloud technologies today (such as Box, Salesforce, and Microsoft Flow) stream data using Base64 encoding. By supporting a Base64-encoded input format, this web service provides greater flexibility to integrate Ephesoft Transact advanced capture capabilities into users' custom solutions.

The v2/ocrClassifyExtractBase64 web service performs OCR, classification and extraction on the supplied input documents. Classification is based only on the first page of each file before extracting the index fields defined in the corresponding batch class. Classification and extraction are performed based on the plugins configured in the corresponding batch class, with two exceptions: 1) The PAGE_PROCESS_SCRIPTING_PLUGIN plugin will not be executed; and 2) Table extraction will be performed if the plugin is turned on, but extracted table data will not be returned in the response.

Only one file can be supplied as input, but multiple files can be processed at one time by combining them in a zip file and submitting the zip file to the web service. However, note that each physical file inside the zip file will be treated as a single logical file, and will be classified based on the contents of the first page only (no separation will occur within the individual files).

This web service supports using a PDF file's EText layer for OCR and extraction, provided that the batch class has been configured accordingly to support that processing method.

The v2/ocrClassifyExtractBase64 web service returns a minimized JSON response.

Here's an example of a JSON input file for the v2/ocrClassifyExtractBase64 web service:

```
{
  "batchClassIdentifier": "BC6",
  "fileName": "Filename.pdf",
  "fileContent": "<Base64-Encoded String>"
}
```

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/v2/ocrClassifyExtractBase64

Image processing web services

convertTiffToPdf

This web service generates PDF files for the input TIFF file. If the input TIFF file is a multi-page file, a multi-page PDF will be generated.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/convertTiffToPdf

Sample client code using Apache Commons HttpClient

```
private static void convertTiffToPdf() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/convertTiffToPdf";
    PostMethod mPost = new PostMethod(url);
    File file1 = new File("C:\\sample\\sample1.tif");
    File file2 = new File("C:\\sample\\sample2.tif");
    Part[] parts = new Part[5];
    try {
        parts[0] = new FilePart(file1.getName(), file1);
        parts[1] = new FilePart(file2.getName(), file2);
        parts[2] = new StringPart("inputParams", "");
        parts[3] = new StringPart("outputParams", "");
        parts[4] = new StringPart("pdfGeneratorEngine", "IMAGE_MAGICK");
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            InputStream in = mPost.getResponseBodyAsStream();
            // output file path for saving results.
            String outputFilePath = "C:\\sample\\serverOutput.zip";
            // retrieving the searchable pdf file
            File f = new File(outputFilePath);
            FileOutputStream fos = new FileOutputStream(f);
            try {
                byte[] buf = newbyte[1024];
```

```
        int len = in .read(buf);
        while (len > 0) {
            fos.write(buf, 0, len);
            len = in .read(buf);
        }
    } finally {
        if (fos != null) {
            fos.close();
        }
    }
} else if (statusCode == 403) {
    System.out.println("Invalid username/password.");
} else {
    System.out.println(mPost.getResponseBodyAsString());
}
} catch (FileNotFoundException e) {
    System.err.println("File not found for processing.");
} catch (HttpException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (mPost != null) {
        mPost.releaseConnection();
    }
}
}
```

createMultiPageFile

This web service will create a multi-page PDF file from one or more single-page TIFF images. A multi-page TIFF file can optionally be created at the same time.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/createMultiPageFile

Sample client code using Apache Commons HttpClient

```
private static void createMultiPage() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/createMultiPageFile";
    PostMethod mPost = new PostMethod(url);
    File file1 = new File("C:\\sample\\WebServiceParams.xml");
    File file2 = new File("C:\\sample\\sample1.tif");
    File file3 = new File("C:\\sample\\sample2.tif");
    Part[] parts = new Part[3];
    try {
        parts[0] = new FilePart(file1.getName(), file1);
        parts[1] = new FilePart(file2.getName(), file2);
        parts[2] = new FilePart(file3.getName(), file3);
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            InputStream inputStream = mPost.getResponseBodyAsStream();
            // Retrieving file from result
            File file = new File("C:\\sample\\serverOutput.zip");
```



```
        FileOutputStream fos = new FileOutputStream(file);
        try {
            byte[] buf = newbyte[1024];
            int len = inputStream.read(buf);
            while (len > 0) {
                fos.write(buf, 0, len);
                len = inputStream.read(buf);
            }
        } finally {
            if (fos != null) {
                fos.close();
            }
        }
        System.out.println("Web service executed successfully..");
    } else if (statusCode == 403) {
        System.out.println("Invalid username/password..");
    } else {
        System.out.println(statusCode + " *** " + mPost.getResponseAsString());
    }
} catch (FileNotFoundException e) {
    System.err.println("File not found for processing..");
} catch (HttpException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (mPost != null) {
        mPost.releaseConnection();
    }
}
}
```

createMultiPagePDFFromSinglePDFs

This web service creates a multi-page PDF file from a number of single page PDF files.

This web service supports using a PDF file's EText layer for OCR and extraction, provided that the batch class has been configured accordingly to support that processing method. If any of the single-page PDF files contain an EText layer, the resulting multi-page PDF file will also contain an EText layer.

Request Method

POST

Web Service URL

<http://<serverName>:<port>/dcma/rest/createMultiPagePDFFromSinglePDFs>

createSearchablePDF

This web service is used to generate a searchable PDF file.

Request Method

POST

Web Service URL

<http://<serverName>:<port>/dcma/rest/createSearchablePDF>

Sample client code using Apache Commons HttpClient

```
private static void createSearchablePDF() {
    HttpClient client = new HttpClient();
    // URL for webservice of create searchable pdf
    String url = "http://localhost:8080/dcma/rest/createSearchablePDF";
    PostMethod mPost = new PostMethod(url);
    File file1 = new File("C:\\sample\\sample1.tif");
    File file2 = new File("C:\\sample\\sample2.tif");
    File file3 = new File("C:\\sample\\sample3.tif");
    File file4 = new File("C:\\sample\\sample4.tif");
    File file5 = new File("C:\\sample\\Fpr.rsp");
    Part[] parts = new Part[9];
    try {
        parts[0] = new FilePart(file1.getName(), file1);
        parts[1] = new FilePart(file2.getName(), file2);
        parts[2] = new FilePart(file3.getName(), file3);
        parts[3] = new FilePart(file4.getName(), file4);
        parts[4] = new FilePart(file5.getName(), file5);
        parts[5] = new StringPart("isColorImage", "false");
        parts[6] = new StringPart("isSearchableImage", "true");
        parts[7] = new StringPart("outputPDFFileName", "OutputPDF.pdf");
        parts[8] = new StringPart("projectFile", "Fpr.rsp");
        parts[9] = new StringPart("ocrEngine", "Recostar");
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            InputStream inputStream = mPost.getResponseBodyAsStream();
            // output file path for saving result
            String outputPath = "C:\\sample\\serverOutput.zip";
            // retrieving the searchable pdf file
            File file = new File(outputPath);
            FileOutputStream fileOutputStream = new FileOutputStream(file);
            try {
                byte[] buf = new byte[1024];
                int len = inputStream.read(buf);
                while (len > 0) {
                    fileOutputStream.write(buf, 0, len);
                    len = inputStream.read(buf);
                }
            } finally {
                if (fileOutputStream != null) {
                    fileOutputStream.close();
                }
            }
            System.out.println("Web service executed successfully.");
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(mPost.getResponseBodyAsString());
        }
    } catch (FileNotFoundException e) {
        System.err.println("File not found for processing.");
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (mPost != null) {
            mPost.releaseConnection();
        }
    }
}
```

}

splitMultipageFile

This web service takes an incoming multi-page PDF or TIFF file and splits it apart into single-page TIFF files. ImageMagick is used for splitting multi-page TIFF files, and Ghostscript is used for splitting multi-page PDF files.

Request Method

POST

Web Service URL

http://<serverName>:<port>/dcma/rest/splitMultipageFile

Sample client code using Apache Commons HttpClient

```
private static void splitMultiPageFile() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/splitMultipageFile";
    PostMethod mPost = new PostMethod(url);
    File file1 = new File("C:\\sample\\sample.pdf");
    File file2 = new File("C:\\sample\\sample.tif");
    Part[] parts = new Part[5];
    try {
        parts[0] = new FilePart(file1.getName(), file1);
        parts[1] = new FilePart(file2.getName(), file2);
        parts[2] = new StringPart("inputParams", "gswin32c.exe -dNOPAUSE -r300 -
sDEVICE=tiff12nc -dBATCH");
        parts[3] = new StringPart("isGhostscript", "true");
        parts[4] = new StringPart("outputParams", "");
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            InputStream in = mPost.getResponseBodyAsStream();
            File file = new File("C:\\sample\\serverOutput.zip");
            FileOutputStream fos = new FileOutputStream(file);
            try {
                byte[] buf = newbyte[1024];
                int len = in.read(buf);
                while (len > 0) {
                    fos.write(buf, 0, len);
                    len = in.read(buf);
                }
            } finally {
                if (fos != null) {
                    fos.close();
                }
            }
            System.out.println("Web service executed successfully..");
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password..");
        } else {
            System.out.println(mPost.getResponseBodyAsString());
        }
    } catch (FileNotFoundException e) {
        System.err.println("File not found for processing..");
    } catch (HttpException e) {
        e.printStackTrace();
    }
}
```

```
} catch (IOException e) {  
    e.printStackTrace();  
} finally {  
    if (mPost != null) {  
        mPost.releaseConnection();  
    }  
}  
}
```

splitMultiPagePDFToSinglePagePDF

This web service splits a multi-page PDF file into single-page PDF files.

This web service supports using a PDF file's EText layer for OCR and extraction, provided that the batch class has been configured accordingly to support that processing method. If any pages of the incoming PDF file contain an EText layer, the single-page PDF files generated from those pages will contain an EText layer.

Request Method

POST

Web Service URL

`http://<serverName>:<port>/dcma/rest/splitMultiPagePDFToSinglePagePDF`

Reporting web services

decryptReportingBatchXml

This web service decrypts the batch.xml file in the report-data folder contained in the SharedFolders area. The web service will find the batch.xml.zip file corresponding to the provided plugin/module name, then decrypt the XML file inside the .zip file.

Request Method

GET

Web Service URL

`http://<serverName>:<port>/dcma/rest/decryptReportingBatchXml/
<batchInstanceIdentifier>/<pluginName>`

Sample client code using Apache Commons HttpClient

```
private static void decryptReportingBatchXml() {  
    HttpClient client = new HttpClient();  
    String url = "http://localhost:8080/dcma/rest/decryptReportingBatchXml/BI2/  
Folder_Import";  
    GetMethod getMethod = new GetMethod(url);  
    int statusCode;  
    try {  
        statusCode = client.executeMethod(getMethod);  
  
        if (statusCode == 200) {  
            System.out.println("Web service executed successfully.");  
            String responseBody = getMethod.getResponseBodyAsString();  
            System.out.println(statusCode + " *** " + responseBody);  
        } else if (statusCode == 403) {
```

```
        System.out.println("Invalid username/password.");
    } else {
        System.out.println(getMethod.getResponseBodyAsString());
    }
} catch (HttpException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (getMethod != null) {
        getMethod.releaseConnection();
    }
}
}
```

runReporting

This web service synchronizes the reporting database.

Request Method

Web Service URL

http://<serverName>:<port>/dcma/rest/runReporting

Sample client code using Apache Commons HttpClient

```
private static void runReporting() {
    HttpClient client = new HttpClient();
    String url = "http://localhost:8080/dcma/rest/runReporting";
    PostMethod mPost = new PostMethod(url);
    File file1 = new File("C:\\sample\\reporting.xml");
    Part[] parts = new Part[1];
    try {
        parts[0] = new FilePart(file1.getName(), file1);
        MultipartRequestEntity entity = new MultipartRequestEntity(parts,
mPost.getParams());
        mPost.setRequestEntity(entity);
        int statusCode = client.executeMethod(mPost);
        if (statusCode == 200) {
            System.out.println("Web service executed successfully.");
            String responseBody = mPost.getResponseBodyAsString();
            System.out.println(statusCode + " *** " + responseBody);
        } else if (statusCode == 403) {
            System.out.println("Invalid username/password.");
        } else {
            System.out.println(mPost.getResponseBodyAsString());
        }
    } catch (FileNotFoundException e) {
        System.out.println("File not found for processing.");
    } catch (HttpException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (mPost != null) {
            mPost.releaseConnection();
        }
    }
}
```

Web services requests

This section provides recommendations to prevent server overload and create request queues to limit the number of web services requests to the Transact server. These recommendations ensure optimal web service performance.


To perform these requests, Transact must be installed with web services enabled.

We recommend implementing a request queue to prevent the server from overloading with web services requests. A request queue is not available in Transact. Users must create a request queue to limit the number of requests to the Transact server.

When requests are sent without a queue, Transact will send the request for processing. This can overload the server if the number of concurrent requests to the Transact server is not controlled within a customer's queue.

The request queue allows a limited number of concurrent requests to the server. Respond to each request individually only upon receiving a response. This is in support of the concurrent requests allowed within a queue. The table below provides recommendations on the number of concurrent requests users can send to Transact based on the CPU cores of the server.

CPU cores	Number of concurrent web services requests	
	Small batches (<10 pages)	Large batches (>50 pages)
4	2	1
8	6	2-3
16	12-14	6-8
32	20-24	10-15

 The values in this table are approximate and may vary according to your use case.

The following information outlines the number of concurrent requests a system can process based on CPU core and web services response time.

- CPU Core: The higher the number of CPU cores, the higher the number of concurrent requests a system can process.
- Web Services Response Time: The least amount of response time needed, the least number of concurrent requests needed to send to the Transact server.

The number of concurrent web services requests should not exceed the number of concurrent batch instances your system can process. Keep the number of web services requests at 25% less than the number of batch instances a system can process.