Kofax Customer Communications Manager

5.0

Calling ITP/Server from JSP

CCM ComposerUI Server 5.0

KOFAX
from Lexmark

Contents

Introduction 3
Copyrights and trademarks 3
ShowLog Example 4
JSP 4
ITP/Server Script 5
Running the Example 5
ListDir Example 6
JSP 6
ITP/Server Script 6
Running the example 7
Example Document Processing 7
JSP 9
ITP/Server Scripts 10
Running the example 10
Custom Tags 11
ServerCall 11
Attributes 11
Type Library 11
ServerParameter 12
Type Library 12
XSLT 12
Type Library 13

Introduction

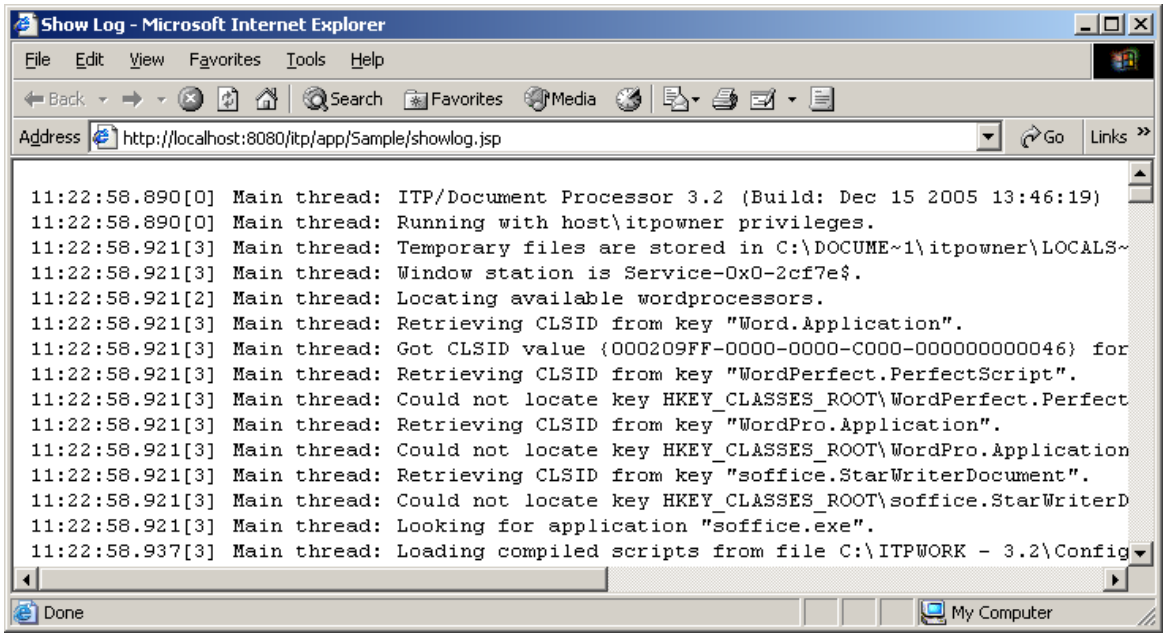
The J2EE version of CCM ComposerUI Server defines a few custom JSP tags that make it very easy to invoke an CCM Core service. Before proceeding with the details, let us have a look at some example JSP pages.

All documentation can also be found on the Kofax Customer Communications Manager Knowledge Center (http://ccmkc.kofax.com).

Copyrights and trademarks

ShowLog Example

The ShowLog example lists the log information of an ITP/Server Document Processor. When invoked, the browser will show the following:



JSP

The following JSP code was needed to implement this. See "showlog.jsp" in the ShowLog directory. Essentially, it contains the following code:

```
<%@ page contentType="text/html" %>
<%@ taglib uri="/WEB-INF/itp.tld" prefix="itp" %>
<html>
<head>
   <meta http-equiv="Content-Type"
        content="text/html; charset=ISO-8859-1">
   <title>Show Log</title>
</head>
<body>
   <pre><itp:ServerCall
       name="showlog" type="online" value="info" /></pre>
</body>
</html>
```

This JSP page uses the "ServerCall" tag to invoke the ITP/Server service "showlog". This service (explained below) returns the log information of the Document Processor that executes the request.

While executing the JSP page, the entire "ServerCall" tag will be replaced by the log data. The resulting HTML page will enclose this data between <pre> tags, e.g:

```
...
<html>
```

```
<head>
   …
</head>

<body>
   <pre> 11:22:58.890[0] Main thread: ITP/Document Processor 3.2 (Build: Dec
15 2005 13:46:19)
11:22:58.890[0] Main thread: Running with host\itpowner privileges
11:22:58.921[3] Main thread: Temporary files are stored in …
11:22:58.921[3] Main thread: Window station is Service-0x0-2 …
```

ITP/Server Script

The service script itself is very basic. It just sends the contents of its local itpdp.log file to ITP/OnLine, identifying it as "info".

```
# ITP/Server script ShowLog

SendFile Dest("info") Src("itpdp.log"[LogDir,]);
```

Running the Example

Running this example takes the following steps:

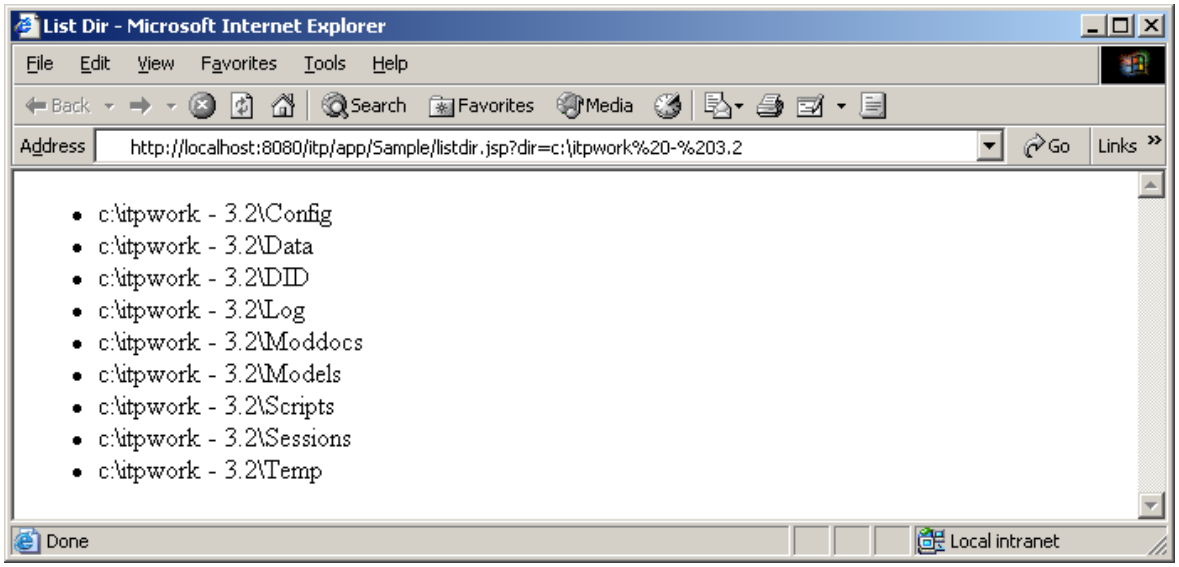
- Create a new ITP/Server service "showlog". Define the script above, compile, and apply.
- Place showlog.jsp in an ITP/OnLine application directory.

You can subsequently invoke the following URL (we placed showlog.jsp in the application "test" and used the machine "host"):

"http://host/itp/app/test/showlog.jsp"

ListDir Example

The ListDir example is slightly more complex. When run with a "dir" parameter, it lists the files at the given directory on the ITP/Server machine.



JSP

The "listdir.jsp" file in the ListDir directory implements this. It includes the following code.

```
<itp:XSLT name="makelist.xsl">
  <itp:ServerCall name="listdir" type="online" value="info">
    <itp:ServerParameter>${param.dir}</itp:ServerParameter>
  </itp:ServerCall>
</itp:XSLT>
```

Here, the "ServerCall" tag calls the ITP/Server service "listdir", passing in the JSP "dir" parameter via a "ServerParameter" sub-element. The ITP/Server listdir service will produce an XML listing of the requested directory. The entire "ServerCall" tag will get replaced by this XML data.

The only thing that remains, is converting the XML data to HTML. ITP/OnLine provides an XSLT tag that applies an XSL-T file to its body. In the JSP page above, the XML list gets transformed by "makelist.xsl" into an HTML list. The body of the resulting HTML code looks as follows.

```
<ul xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<li>c:\itpwork - 3.2\Config</li>
<li>c:\itpwork - 3.2\Data</li>
<li>c:\itpwork - 3.2\DID</li>
<li>c:\itpwork - 3.2\Log</li>
…
</ul>
```

ITP/Server Script

The following ITP/Server script generates an XML listing of the directory that is specified in its Dir parameter.

```
# ITP/Server script ListDir

Parameter Text Version;
Parameter Text Dir;

Const Text List = "list.xml"[TempDir,];
Temporary File(List);

WriteFile File(List) Message("<items>");
Var Text File;
ForEach File Folder Dir Do
   WriteFile File(List) Message("<item>");
   WriteFile File(List) Message("<![CDATA[" + File + "]]>");
   WriteFile File(List) Message("</item>");
Od;
WriteFile File(List) Message("</items>");

SendFile Dest("info") Src(List);
```

Notice that this script specifies an additional "Version" parameter. All ServerCall invocations of type "online" implicitly get this extra parameter. It specifies the version of the low-level protocol that ITP/OnLine uses. Currently, it is "1", and there is no need to take it into account[2].

Running the example

Running this example takes the following steps:

- Create a new ITP/Server service "listdir". Define the script above.
- Before compiling the listdir service, make sure to define Version=$1 and Dir=$2 for the service parameters.
- Place listdir.jsp in an ITP/OnLine application directory.
- Place "makelist.xsl" in the same application directory as listdir.jsp.

You can subsequently invoke the following URL (we placed listdir.jsp in the application "test" and used the machine "host"):

"http://host/itp/app/test/showlog.jsp?dir=c:/"

The dir parameter ('dir=') should be used to specify the directory to view.

Example Document Processing

A more realistic example involves processing a document that has been generated by ITP/OnLine, typically by employing a modified modelend.jsp. By default, modelend.jsp just opens the document in the browser. Below, we will show an example that lets the user choose to either:

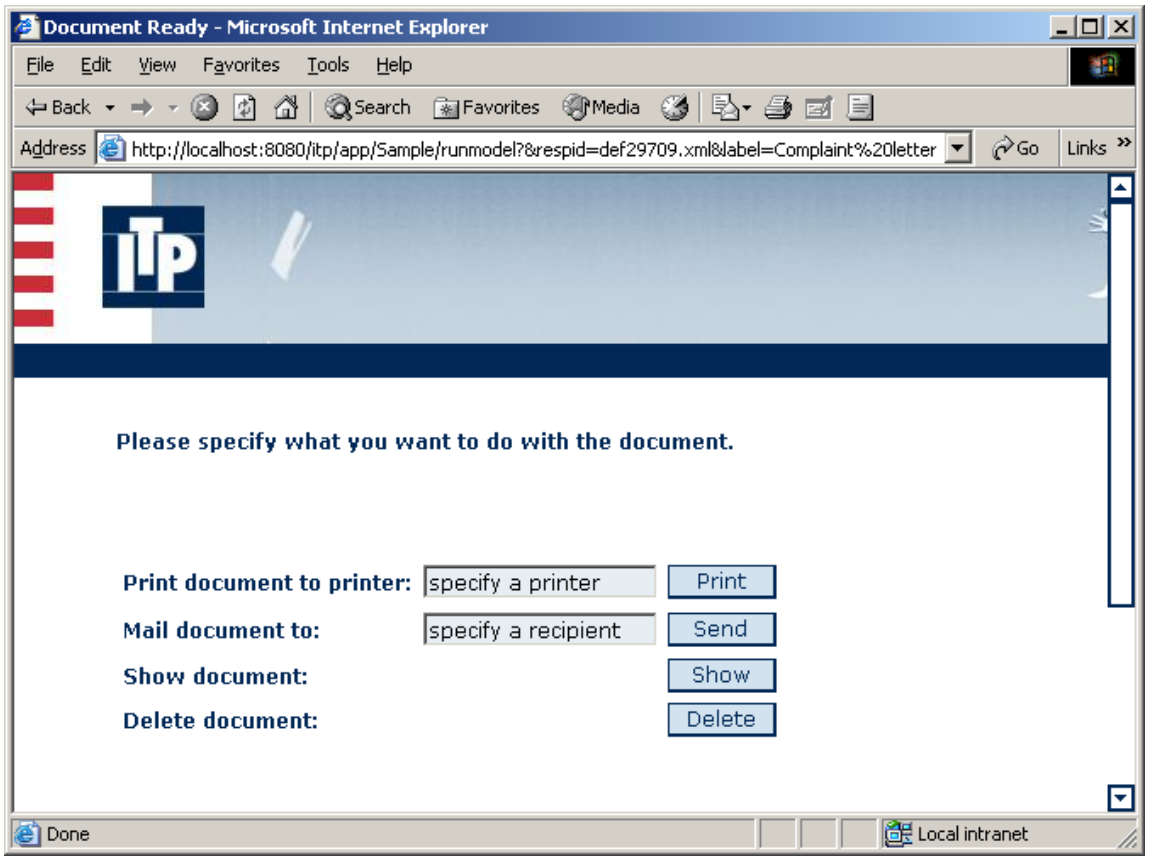
- Print the document to a printer.
- Mail the document as an attachment to an e-mail address.
- Show the document in a browser window.

- Delete the document.

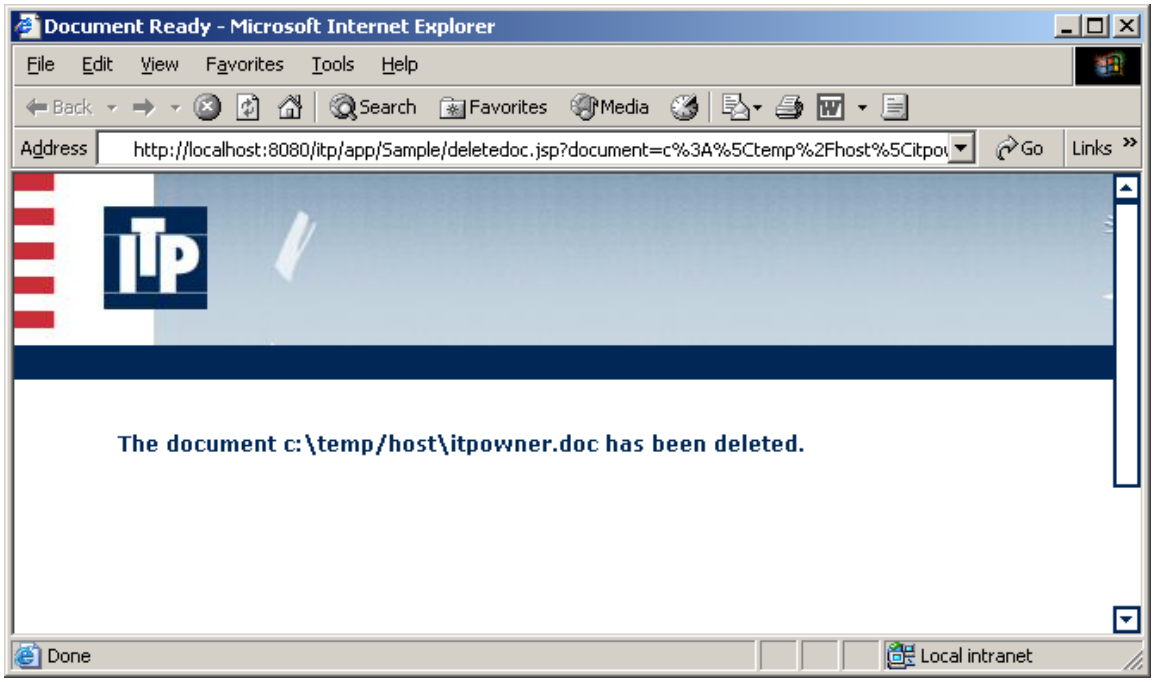
This example explicitly specifies that ITP should generate the document in a location on the ITP/Server machine. This location must be passed as a 'respath' argument on the URL, for example:

```
http://host/itp/app/test/modelselect.jsp?respath=c:/temp
```

This example will start as a normal ITP/OnLine run. After selecting and completing a model run, the following page will be presented:



After choosing print, send or show, you will return to this page, so you can choose additional actions. After choosing delete, you will reach this page.



JSP

This example involves 3 JSP pages:

- Modelbegin.jsp. This is an override of the normal modelbegin.jsp. It specifies a location for the final document that is unique for each user. Before calling runmodel, it adds three parameters:
  - Res_srv is set to "y" so that ITP/OnLine stores the document on the ITP/Server machine.
  - Res_owt is set to "y" so that existing documents get overwritten.
  - Res_uri is set to <respath>/<domain>/<user>.doc. It uses the itp:user variable for this, which is stored at session level. This variable delivers the current user id as <domain>\<user>.
- Modelend.jsp. This is an override of the normal modelend.jsp page. It outputs the processing selection page, and also calls ITP/Server to execute print, mail and show.
- Deletedoc.jsp. This page outputs the final 'delete' page, and also implements the delete action.

The last two pages are too long to list or discuss here. See the jsp pages in the DocProc directory for details.

There is however one new thing in these JSP pages that needs mentioning: the call to the ITP/Server service DocProcGet in modelend.jsp specifies a different value attribute:

```
<itp:ServerCall name="DocProcGet" type="online" value="document">
  <itp:ServerParameter value="${param.show}"/>
</itp:ServerCall>
```

This type of value specifies that the ServerCall tag will be replaced by a URL to a *copy* of the document.

ITP/Server Scripts

The four document actions each have been implemented by their own ITP/Server script. See DocProc/scripts.

- DocProcPrint.dss. Prints the document.
- DocProcMail.dss. Mails the documents.
- DocProcGet: retrieves the document and sends it to ITP/OnLine (which then picks it up and shows it).
- DocProcDelete: Deletes the document.

All these scripts are fairly short and self-explanatory.

Running the example

- Copy the ITP/Server scripts in the DocProc/scripts directory to the ITP/Server scripts directory and define the corresponding services, and map the parameters in the order that they are listed in the scripts (Version=$1, etc.). Compile and apply.
- Copy the jsp pages and the css directory to an ITP/OnLine application directory. The css directory just contains definitions for fonts, images, etc., which is of no concern here.

Custom Tags

The following custom tags have been defined by ITP/OnLine. You can find their type library definitions in WEB-INF/itp.tld.

ServerCall

The ServerCall tag invokes an ITP/Server service. Depending on its attributes (see below) the ServerCall tag itself will get replaced by the value/result of this invocation.

A general invocation looks as follows:

```
<itp:ServerCall name="…" …>
  <itp:ServerParameter>…</itp:ServerParameter>
  …
  <itp:ServerParameter>…</itp:ServerParameter>
</itp:ServerCall>
```

The JSP code above will invoke the service with the specified name, and pass the parameters in the order that they are listed. For details, see below.

Attributes

- name: the name of the ITP/Server service (required).
- type: the type of invocation. Currently, this should be "online".
- value: if specified, the ServerCall tag will be replaced by the indicated value.
  Currently, only two values are supported for the value attribute:
  - info: this means that the last data sent by SendFile Dest("info") … will be used to replace the tag.
  - document: a URI will be returned to a *copy* of the document that was sent by SendFile Dest ("document.<extension>"). Each call will deliver a *new copy*, as multiple calls may not deliver the same document (this depends on the script). These copies get stored in a location that is related to the browser session, so only after the browser session expires these documents will be removed. One should *not* write JSP code that depends on the particular form of URI that gets returned.

If the value attribute is not specified, or if it is not supported, the ServerCall tag does not return a result.

- parameters: a comma-separated list of parameters. This offers an alternative way for specifying ITP/Server parameters. In particular this is useful for parameter lists of unknown size that originate from web forms.
- sessionid: identifier of the ITP/Server session in which this job will run

Type Library

The type library definition looks as follows:

```
<tag>
  <description>Call ITP Server</description>
  <name>ServerCall</name>
  <tag-class>…</tag-class>
  <body-content>JSP</body-content>
  <attribute>
    <name>name</name>
    <required>true</required>
    <rtexprvalue>true</rtexprvalue>
  </attribute>
  <attribute>
    <name>type</name>
    <required>false</required>
    <rtexprvalue>true</rtexprvalue>
  </attribute>
  <attribute>
    <name>value</name>
    <required>false</required>
    <rtexprvalue>true</rtexprvalue>
  </attribute>
  <attribute>
    <name>parameters</name>
    <required>false</required>
    <rtexprvalue>true</rtexprvalue>
  </attribute>
</tag>
```

ServerParameter

The ServerParameter tag is used to pass parameter values to an enclosing ServerCall tag. Each ServerParameter element can specify a value in one of two ways (the first one taking precedence):

- <ServerParameter>…</ServerParameter>
- <ServerParamerer value="…" />

Type Library

```
<tag>
  <description>Set ITP Server call parameter</description>
  <name>ServerParameter</name>
  <tag-class>…</tag-class>
  <body-content>scriptless</body-content>
  <attribute>
    <name>value</name>
    <required>false</required>
    <rtexprvalue>true</rtexprvalue>
  </attribute>
</tag>
```

XSLT

The XSLT tag applies an XSL-T transformation on the body of the tag, and replaces the tag by the result of this transformation.

It is a single "name" attribute that specifies an XSL-T filename. This filename is relative to the JSP page.

Type Library

```
<tag>
  <description>
    Perform an xslt transformation on the body
  </description>
  <name>XSLT</name>
  <tag-class>com.aia_itp.itpols.frontend.xsltTag</tag-class>
  <body-content>JSP</body-content>
  <attribute>
    <name>name</name>
    <required>true</required>
    <rtexprvalue>false</rtexprvalue>
  </attribute>
</tag>
```