Kofax Communications Manager

ComposerUI for J2EE Developer's Guide

Version: 5.3.0

Date: 2019-05-28

KOFAX

Legal Notice

Table of Contents

Preface

The J2EE version of KCM ComposerUI Server defines a few custom JSP tags that simplify the process of invoking the KCM Core service. This guide describes JSP pages and lists custom tags defined by KCM ComposerUI for J2EE.

Related documentation

The documentation set for Kofax Communications Manager is available here:[1]

https://docshield.kofax.com/Portal/Products/CCM/530-1h4cs6680a/CCM.htm

The documentation set includes the following items:

Kofax Communications Manager Release Notes
Contains late-breaking details and other information that is not available in your other Kofax Communications Manager documentation.

Kofax Communications Manager Batch & Output Management Getting Started Guide
Describes how to start working with Batch & Output Management.

Kofax Communications Manager Getting Started Guide
Describes how to use Contract Manager to manage instances of Kofax Communications Manager.

Help for Kofax Communications Manager Designer
Contains general information and instructions on using Kofax Communications Manager Designer, which is an authoring tool and content management system for Kofax Communications Manager.

Kofax Communications Manager Repository Administrator's Guide
Describes administrative and management tasks in Kofax Communications Manager Repository and Kofax Communications Manager Designer for Windows.

Kofax Communications Manager Repository User's Guide
Includes user instructions for Kofax Communications Manager Repository and Kofax Communications Manager Designer for Windows.

[1] You must be connected to the Internet to access the full documentation set online. For offline access, see the "Product documentation" section in the *Installation Guide*.

Kofax Communications Manager Repository Developer's Guide
Describes various features and APIs to integrate with Kofax Communications Manager Repository and Kofax Communications Manager Designer for Windows.

Kofax Communications Manager Template Scripting Language Developer's Guide
Describes the KCM Template Script used in Master Templates.

Kofax Communications Manager Core Developer's Guide
Provides a general overview and integration information for Kofax Communications Manager Core.

Kofax Communications Manager Core Scripting Language Developer's Guide
Describes the KCM Core Script.

Kofax Communications Manager API Guide
Describes Contract Manager, which is the main entry point to Kofax Communications Manager.

Kofax Communications Manager ComposerUI for HTML5 JavaScript API Web Developer's Guide
Describes integration of ComposerUI for HTML5 into an application, using its JavaScript API.

Kofax Communications Manager DID Developer's Guide
Provides information on the Database Interface Definitions (referred to as DIDs), which is a deprecated method to retrieve data from a database and send it to Kofax Communications Manager.

Kofax Communications Manager ComposerUI for ASP.NET and J2EE Customization Guide
Describes the customization options for KCM ComposerUI for ASP. NET and J2EE.

Kofax Communications Manager ComposerUI for ASP.NET Developer's Guide
Describes the structure and configuration of KCM ComposerUI for ASP. NET.

Getting help with Kofax products

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

To access the Kofax Knowledge Base, go to the Kofax website and select **Support** on the home page.

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

The Kofax Knowledge Base 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.

  Scroll through the Kofax Knowledge Base home page to locate a product family. Then click a product family name to view a list of related articles. Please note that some product families require a valid Kofax Portal login to view related articles.
- Access to the Kofax Customer Portal (for eligible customers).

  Click the **Customer Support** link at the top of the page, and then click **Log in to the Customer Portal**.
- Access to the Kofax Partner Portal (for eligible partners).

  Click the **Partner Support** link at the top of the page, and then click **Log in to the Partner Portal**.
- Access to Kofax support commitments, lifecycle policies, electronic fulfillment details, and self-service tools.

  Scroll to the **General Support** section, click **Support Details**, and then select the appropriate tab.

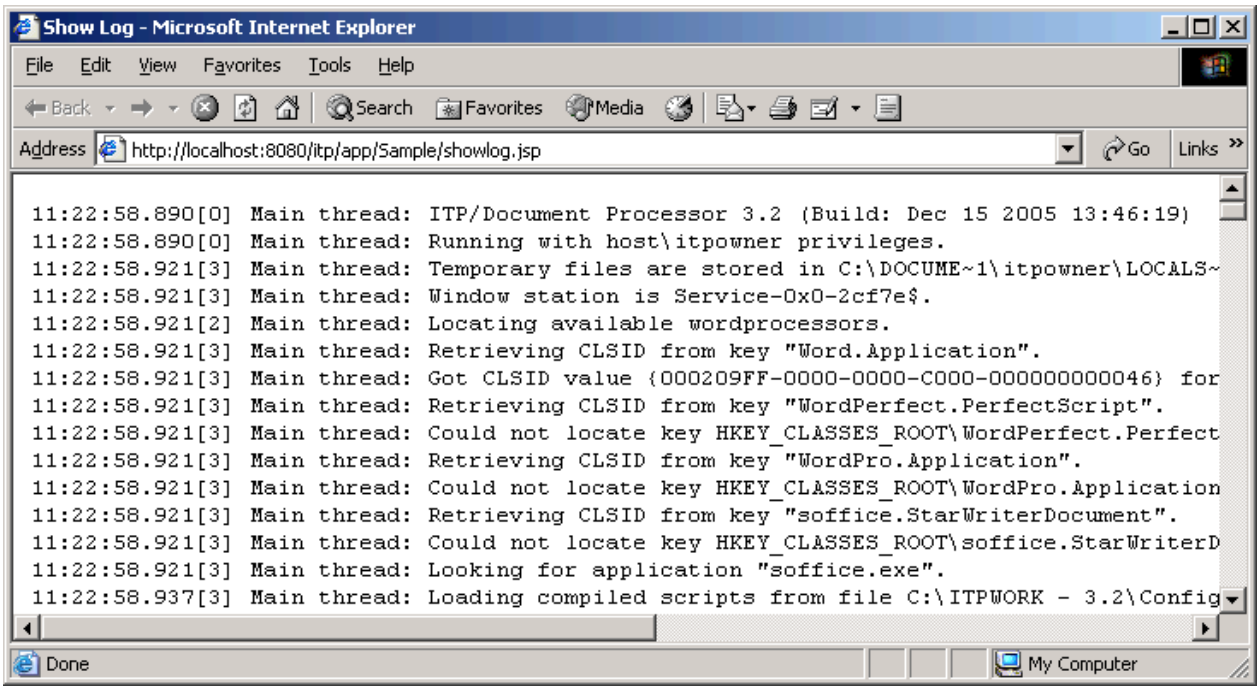
Chapter 1

J2EE examples

In this chapter, you will find descriptions of JSP pages, along with a detailed explanation of several J2EE examples.

ShowLog example

The ShowLog example lists the log information of the KCM Core Document Processor. When ShowLog is invoked, the log information appears in the browser, as shown here.



JSP

The following JSP code is required for implementation. See "showlog.jsp" in the ShowLog directory.

```
<%@ 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 KCM Core service "showlog." This service returns the log information of the Document Processor that executes the request.

While executing the JSP page, the entire "ServiceCall" tag is replaced by the log data. The resulting HTML page encloses this data within <pre> tags. For example:

```
…
<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 …
```

KCM Core script

The service script is a basic command that sends the contents of its local itpdp.log file to the KCM Core, identifying it as "info."

```
# ITP/Server script ShowLog

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

Run the Example

Running this example includes the following steps:

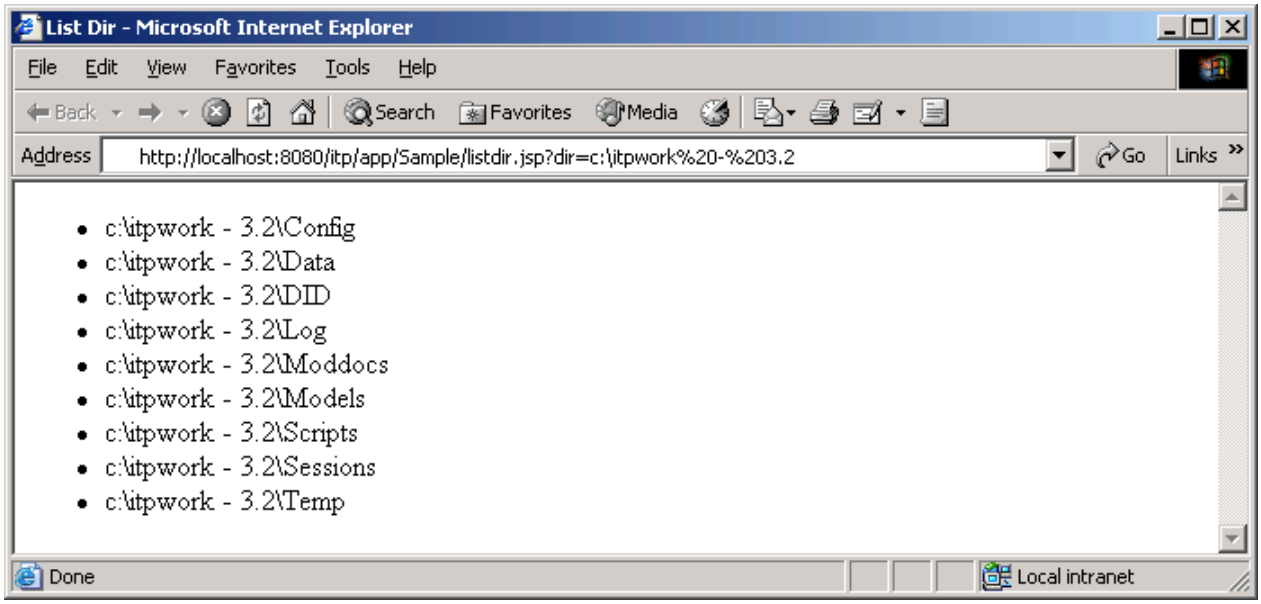
- Create a new KCM Core service for "showlog." Define the script above, compile and apply.
- Place showlog.jsp in the KCM Core application directory.

You can subsequently invoke the following URL, where showlog.jsp is placed in the application "test" and the machine "host" is used:

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

ListDir example

The ListDir example is slightly more complex than the SnowLog Example. When run with a "dir" parameter, it lists the files at the given directory on the KCM Core machine.



JSP

The following JSP code is implemented by the "listdir.jsp" file in the ListDir directory:

```
<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 KCM Core service "listdir," passing in the JSP "dir" parameter via a ServerParameter sub-element. The KCM Core listdir service produces an XML listing of the requested directory. The entire ServerCall tag gets replaced by this XML data.

The only thing that remains the same is converting the XML data to HTML. KCM Core provides an XSLT tag that applies an XSLT file to its body. In the JSP page above, the XML list is transformed by "makelist.xsl" into an HTML list. The body of the resulting HTML code appears 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>
```

KCM Core script

The following KCM Core 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);
```

Note 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 KCM Core uses. Currently, it is 1, and there is no need to take it into account 2.

Run the example

Running this example involves the following steps:

- Create a new KCM Core 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 the KCM Core application directory.
- Place "makelist.xsl" in the same application directory as listdir.jsp.

You can subsequently invoke the following URL, where listdir.jsp is placed in the application "test" and the machine "host" is used:

```
http://host/itp/app/test/listdir.jsp?dir=c:/
```

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

Document processing example

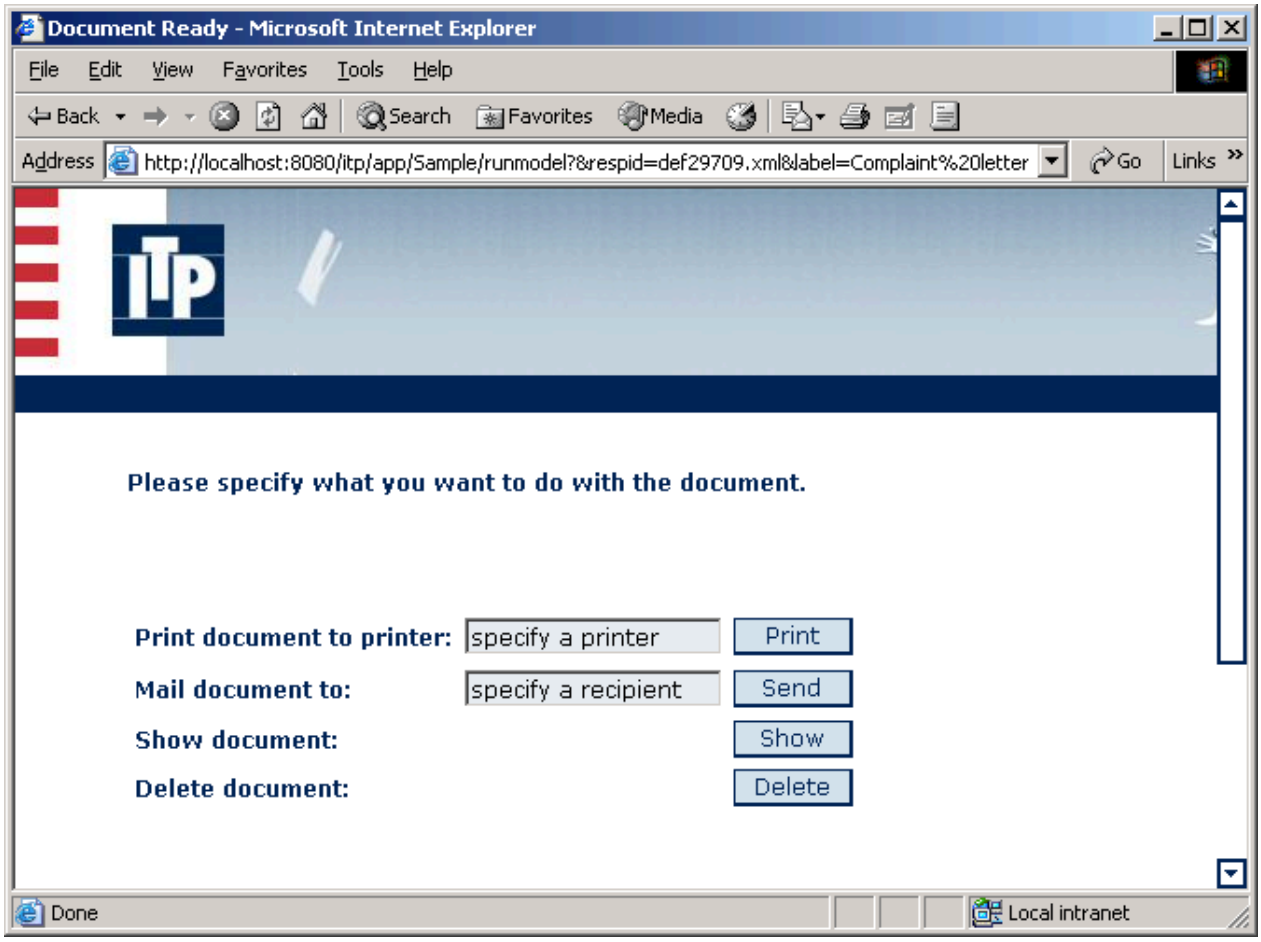
A more practical example involves processing a document that is generated by KCM ComposerUI, typically by employing a modified modelend.jsp. By default, modelend.jsp just opens the document in the browser. As the example below shows, the user can select available actions.

- Print the document to a printer.
- Mail the document as an email attachment.
- Show the document in a browser window.
- Delete the document.

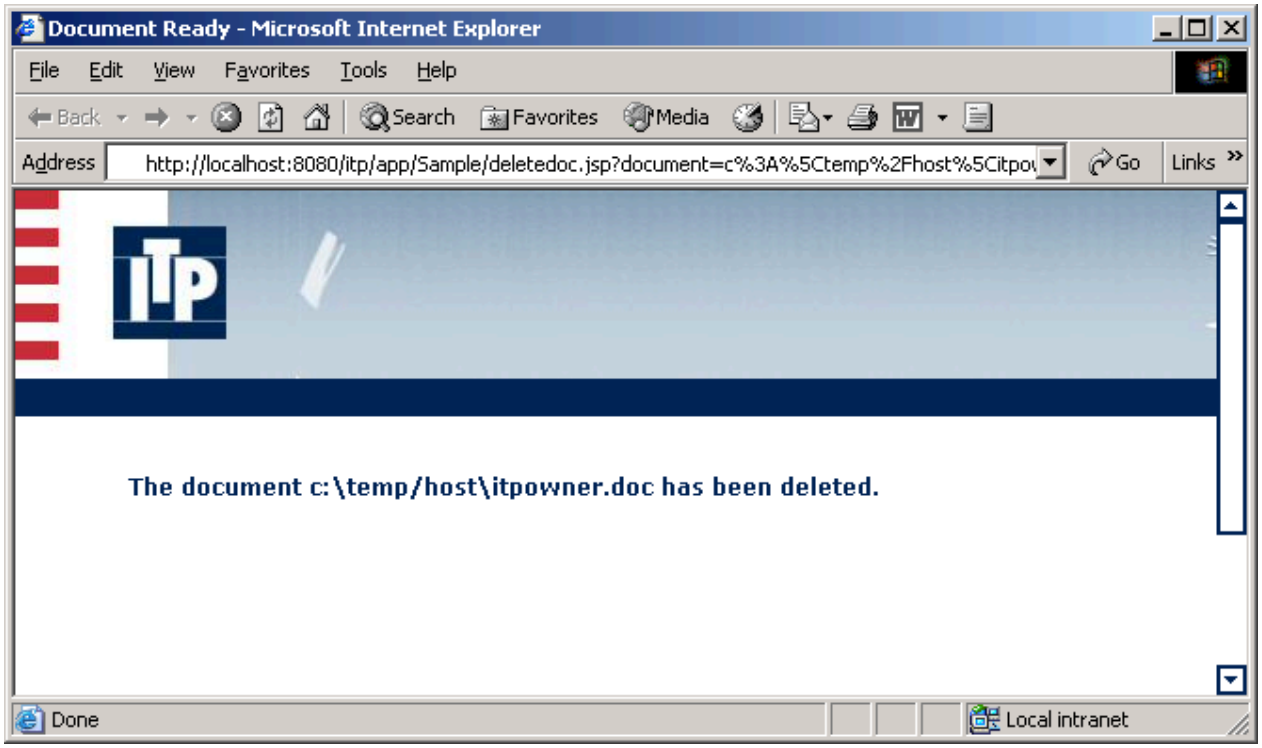
This example explicitly specifies that KCM should generate the document in a location on the KCM Core machine. This location must be passed as a "respath" argument on the URL, such as:

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

This example starts as a normal KCM ComposerUI run. After selecting and completing a model run, the following page appears.



After clicking Print, Send or Show, you are returned to this page to select additional options. After clicking Delete, you reach this page.



JSP

This example involves three JSP pages:

- Modelbegin.jsp. This is an override of the standard 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 KCM ComposerUI stores the document on the KCM Core machine.
  - Res_owl 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, which is stored at the session level. This variable delivers the current user ID as <domain>\<user>.
- Modelend.jsp. This is an override of the standard modelend.jsp page. It outputs the processing selection page, and also calls KCM Core to execute print, mail and show.
- Deletedoc.jsp. This page outputs the final "delete" page, and also implements the delete action.

See the JSP pages in the DocProc directory for more information.

The call to the KCM Core 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 is replaced by a URL to a `copy` of the document.

KCM Core scripts

Each of the following four document actions is implemented by its own KCM Core script. See DocProc/scripts.

- DocProcPrint.dss. Prints the document.
- PocProcMail.dss. Mails the documents.
- DocProcGet. Retrieves the document and sends it to KCM ComposerUI, which picks it up and shows it.
- DocProcDelete. Deletes the document.

All these scripts are short and self-explanatory.

Run the example

- Copy the KCM Core scripts in the DocProc/scripts directory to the KCM Core scripts directory and define the corresponding services. Map the parameters in the same order they are listed in the scripts (Version=$1, etc.). Compile and apply.
- Copy the JSP pages and the CSS directory to the KCM ComposerUI application directory. The CSS directory only contains definitions for fonts, images and other elements.

Chapter 2

Custom Tags

This chapter lists custom tags are defined by KCM ComposerUI for J2EE. You can find their type library definitions in WEB-INF/itp.tld.

ServerCall

The ServerCall tag invokes a KCM Core service. Depending on its attributes presented below, the ServerCall tag itself gets replaced by the value/result of this invocation.

A general invocation appears as follows:

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

The JSP code above invokes the service with the specified name and passes the parameters in the order they are listed. See the next section.

Attributes

- name. The name of the KCM Core service. This is a required attribute.
- type. The name of invocation. Currently, this should be "online."
- value. If specified, the ServerCall tag is 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") is used to replace the tag.
  - document. A URL is returned to a *copy* of the document that was sent by SendFile Dest ("document.<extension>"). Each call delivers the same document, which depends on the script. These copies are stored in a location related to the browser session. These documents are removed only after the browser session is expired. It is `not` recommended to write JSP code depending on the particular form of URI that is returned.

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

- parameters. A comma-separated list of parameters. This offers an alternate way to specify KCM Core parameters. This is particularly useful for parameter lists of unknown size that originate from web forms.
- sessionid. Identifier of the KCM Core session in which this job runs.

Type library

The Type library definitions appears 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 the enclosing ServerCall tag. Each ServerParameter element can specify a value in one of the following two ways, where the first one is taking precedence:

- <ServerParameter>…</ServerParameter>
- <ServerParameter 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 XSLT 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 XSLT 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>
```