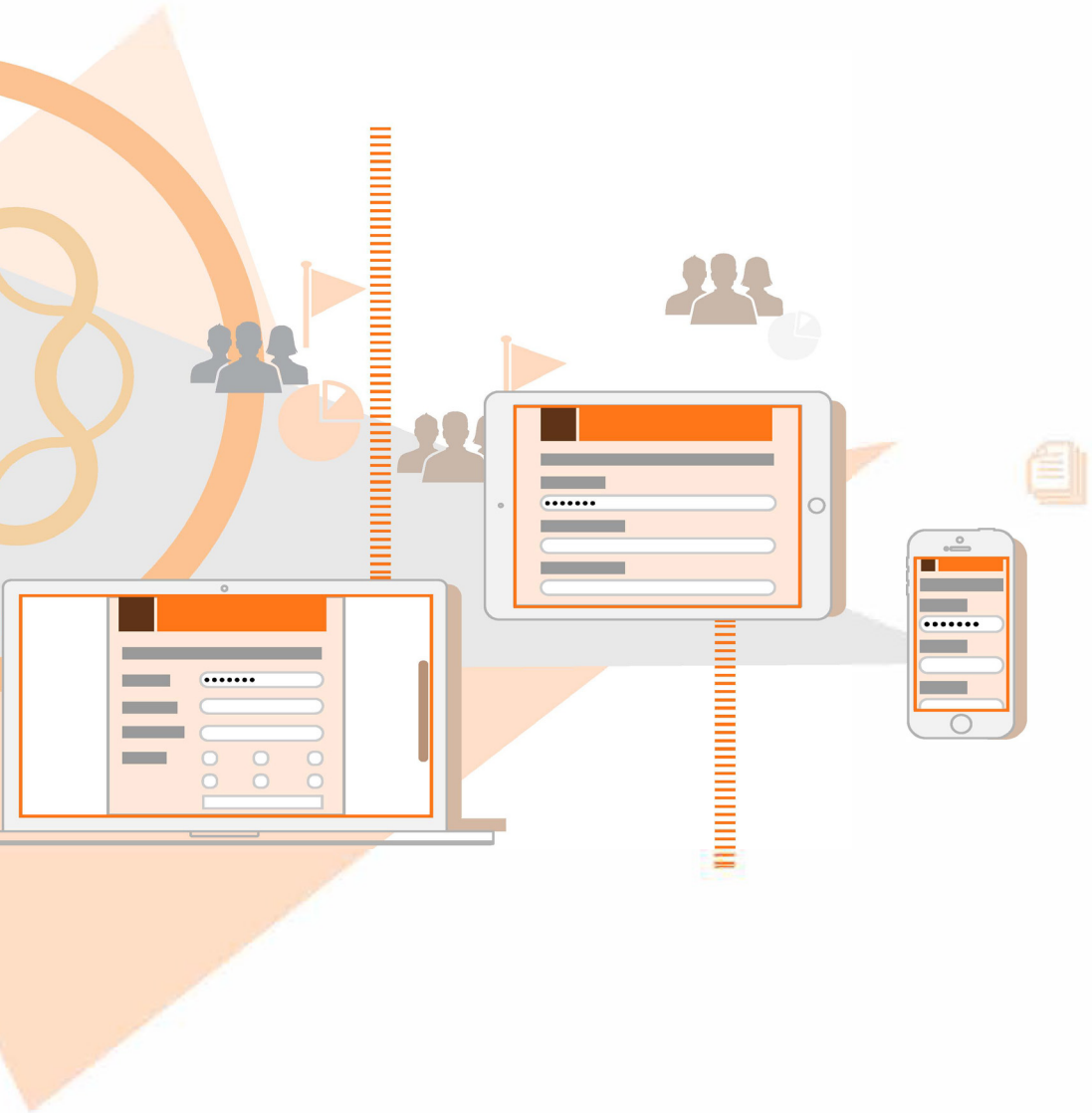
AEM Forms on JEE Services Reference



AEM 6.5 Forms

Legal notices

Contents

1. About This Document

This document describes the services that developers can use to create AEM Forms applications.

1.1. What's in this document

This document provides introductory information about services in AEM Forms on JEE and how they can be used to accomplish different tasks as part of a business process. It also includes information about managing services and applications.

1.2. Who should read this document

This document is primarily intended for people who are designing processes in workbench. It is intended for developers who want to build client applications that programmatically interact with services. The document also includes information of interest to administrators who manage forms servers and applications.

1.3. Additional information

In addition to this document, the resources in the table provide further information about the services.

For information about	See
Using a service in a process map	WorkbenchHelp
Using a service's API	ProgrammingwithAEM Forms
Administering a service	AdministrationConsoleHelp

2. Introducing Services for AEM Forms on JEE

Each AEM Forms on JEE module uses several services to accomplish different tasks as part of a business process. This document describes the services that developers and programmers can use to create AEM Forms on JEE applications.

2.1. Determining which services belong to a module

Each service is licensed for use with one or more AEM Forms on JEE modules in a production environment. The AEM Forms on JEE Foundation services are licensed for use with all AEM Forms modules.

2.2. How developers interact with services

When a service or application is deployed within AEM Forms on JEE, it can be invoked by a client application by using different mechanisms. From the administration console, a service can be configured to be exposed by one or more of these mechanisms.

You can interact with a service in any of the following ways:

- Develop a process in workbench that uses the service.
- Develop a client application that uses the service's API in Java™ or in an environment that permits you to use its exposed WSDL, such as Microsoft® Visual Studio .NET. Most services provide public APIs.
- Develop a client application in Adobe Flash® Builder™ that uses (Deprecated for AEM Forms) AEM Forms on JEE Remoting to interact with the service. Most services can be invoked by using (Deprecated for AEM Forms) AEM Forms on JEE Remoting.

  A service can also be invoked by using email and watched folders. Additionally, some services can automatically invoke other services. The Assembler service can invoke other services that are specified in the DDX file it processes. The Generate PDF service can invoke other services that are specified by the PDFG Internet Printer feature.

  For more information about the different ways to programmatically invoke services, see "Invoking AEM Forms on JEE" in Programming with AEM Forms.

2.3. How administrators interact with services

You can use administration console to perform these tasks:

- Configure and manage users, groups, and server authentication settings by using User Management.
- Create and manage invocation endpoints and deploy AEM Forms on JEE archive (LCA) files.
- Set up watched folders and email providers for non-programmatic process invocation.

- Administer module properties and server settings such as port numbers and log files.

3. Assembler Service

3.1. About DDX

When using the Assembler service, use an XML-based language called *Document Description XML* (DDX) to describe the output you want. DDX is a declarative markup language whose elements represent building blocks of documents. These building blocks include PDF documents, XDP documents, XDP form fragments, and other elements such as comments, bookmarks, and styled text.

DDX document can specify resultant documents with these characteristics:

- PDF document that is assembled from multiple PDF documents
- Multiple PDF documents that are broken apart from a single PDF document
- PDF Portfolio that includes a self-contained user interface and multiple PDF and non-PDF documents
- XDP document that is assembled from multiple XDP documents
- XDP document that contains XML fragments that are dynamically inserted into an XDP document
- PDF document that packages an XDP document
- XML files that report on the characteristics of a PDF document. The reported characteristics include text, comments, form data, file attachments, files used in PDF Portfolios, bookmarks, and PDF properties. PDF properties include form properties, page rotation, and document author.

You can use DDX to augment PDF documents as part of document assembly or disassembly. You can specify any combination of the following effects:

- Add or remove watermarks or backgrounds on selected pages.
- Add or remove headers and footers on selected pages.
- Removes the structure and navigational capabilities of a PDF Package or PDF Portfolio. The result is a single PDF file.
- Renumber page labels. Page labels are typically used for page numbering.
- Import metadata from another source document.
- Add or remove file attachments, bookmarks, links, comments, and JavaScript.
- Set initial view characteristics and optimize for viewing on the web.
- Set permissions for encrypted PDF.
- Rotate pages or rotate and shift content on pages.
- Change the size of selected pages.
- Merge data with a PDF that is XFA-based.

You can use a simple input map to specify the locations of source and resultant documents. You can also use the following external data URL types:

- Application
- Contentspace (deprecated)
- File
- FTP
- HTTP/HTTPS
- Inputmap
- Process
- Repository (deprecated for use in the Assembler service)

3.2. Using the Assembler service

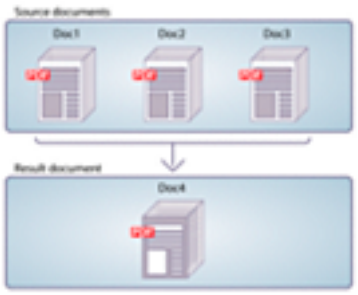
For information about developing processes that use this service, see Workbench Help. For information about developing client applications that programmatically interact with this service, see Programming with AEM Forms.

Assemble PDF documents

You can use the Assembler service to assemble two or more PDF documents into a single PDF document or PDF Portfolio. You can also apply features to the PDF document that aid navigation or enhance security. Here are some of the ways you can assemble PDF documents:

Assemble a simple PDF document

The following illustration shows three source documents being merged into a single resultant document.



The following example is a simple DDX document used to assemble the document. It specifies the names of the source documents used to produce the resultant document, as well as the name of the resultant document:

```
<PDF result="Doc4">
<PDF source="Doc1"/>
<PDF source="Doc2"/>
<PDF source="Doc3"/>
</PDF>
```

Document assembly produces a resultant document that contains the following content and characteristics:

- All or part of each source document
- All or part of the bookmarks from each source document, normalized for the assembled resultant document
- Other characteristics adopted from the base document (Doc1), including metadata, page labels, and page size
- Optionally, the resultant document includes a table of contents constructed from the bookmarks in the source documents

Create a PDF Portfolio

The Assembler service can create PDF Portfolios that contain a collection of documents and a self-contained user interface. The interface is called a *PDF Portfolio Layout* or a *PDF Portfolio navigator* (navigator). PDF Portfolios extend the capability of PDF packages by adding a navigator, folders, and welcome pages. The interface can enhance the user experience by taking advantage of localized text string, custom color schemes, and graphic resources. The PDF Portfolio can also include folders for organizing the files in the portfolio.

When the Assembler service interprets the following DDX document, it assembles a PDF Portfolio that includes a PDF Portfolio navigator and a package of two files. The service obtains the navigator from the location specified by the `myNavigator` source. It changes the navigator's default color scheme to the `pinkScheme` color scheme.

```
<DDX xmlns="http://ns.adobe.com/DDX/1.0/">
 <PDF result="Untitled 1">
   <Portfolio>
     <Navigator source="myNavigator"/>
     <ColorScheme scheme="pinkScheme"/>
   </Portfolio>
   <PackageFiles>
     <PDF source="sourcePDF1"/>
     <PDF source="sourcePDF2"/>
   </PackageFiles>
 </PDF>
</DDX>
```

Assemble encrypted documents

When you assemble a document, you can also encrypt the PDF document with a password. After a PDF document is encrypted with a password, a user must specify the password to view the PDF document in Adobe Reader or Acrobat. To encrypt a PDF document with a password, the DDX document must contain encryption element values that are required to encrypt a PDF document.

The Encryption service does not have to be part of your AEM Forms on JEE installation to encrypt a PDF document with a password.

If one or more of the input documents is encrypted, provide a password to open the document as part of the DDX.

Assemble documents using Bates numbering

When you assemble a document, you can use Bates numbering to apply a unique page identifier to each page. When you use Bates numbering, each page in the document (or set of documents) is assigned a number that uniquely identifies the page. For example, manufacturing documents that contain bill of material information and are associated with the production of an assembly can contain an identifier. A Bates number contains a sequentially incremented numeric value and an optional prefix and suffix. The prefix + numeric value + suffix is called a *bates pattern*.

The following illustration shows a PDF document that contains a unique identifier located in the document's header.

Flatten and assemble documents

You can use the Assembler service to transform an interactive PDF document (for example, a form) to a non-interactive PDF document. An interactive PDF document lets users enter or modify data located in the PDF document fields. The process of transforming an interactive PDF document to a non-interactive PDF document is called *flattening*. When a PDF document is flattened, form fields retain their graphical appearance but are no longer interactive. One reason to flatten a PDF document is to ensure that data cannot be modified. In addition, scripts associated with the fields no longer function.

NOTE: The Assembler service uses the Output service to flatten dynamic XFA forms. If the Assembler service processes a DDX that requires it to flatten an XFA dynamic form and the Output service is unavailable, an exception is thrown. The Assembler service can flatten an Acrobat form or a static XFA form without using the Output service.

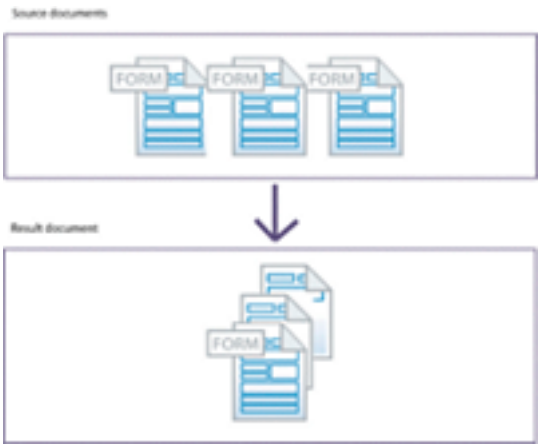
Assemble XDP documents

You can use the Assembler service to assemble multiple XDP documents into a single XDP document or into a PDF document. For source XDP files that include insertion points, you can specify the fragments to insert.

Here are some of the ways you can assemble XDP documents:

Assemble a simple XDP document

The following illustration shows three source XDP documents being assembled into a single resultant XDP document. The resultant XDP document contains the three source XDP documents including their associated data. The resultant document obtains basic attributes from the base document, which is the first source XDP document.



Here is a DDX document that produces the result illustrated above.

```
<DDX xmlns="http://ns.adobe.com/DDX/1.0/">
 <XDP result="MyXDPResult">
   <XDP source="sourceXDP1"/>
   <XDP source="sourceXDP2"/>
   <XDP source="sourceXDP3"/>
 </XDP>
</DDX>
```

Resolving references during assembly

Typically, XDP documents can contain images referenced either through absolute or relative references. Assembler service, by default, retains the references to the images in the resultant XDP document.

You can specify how the Assembler service handles the images referenced in the source XDP documents either through absolute or relative references in the XDP files when assembling. You can choose to have all the images embedded in the resultant so that it contains no relative or absolute references. You define this by setting the value of the `resolveAssets` tag, which can take any of the following options. By default, no references are resolved in the result document.

Value	Description
none	Does not resolve any references.
all	Embeds all referenced images in the source XDP document.
relative	Embeds all the images referenced through relative references in the source XDP document.
absolute	Embeds all the images referenced through absolute references in the source XDP document.

You can specify the value of the `resolveAssets` attribute either in the XDP source tag or in the parent XDP result tag. If the attribute is specified to the XDP result tag, it will be inherited by all the XDP source elements which are children of XDP result. However, explicitly specifying the attribute for a source element overrides the setting of the result element for that source document alone.

Resolve all source references in an XDP document

To resolve all references in the source XDP documents, specify the `resolveAssets` attribute for the resultant document to *all*, as in the example below:

```
<DDX xmlns="http://ns.adobe.com/DDX/1.0/">
<XDP result="result.xdp" resolveAssets="all">
<XDP source="input1.xdp" />
<XDP source="input2.xdp" />
<XDP source="input3.xdp" />
</XDP>
</DDX>
```

You can also specify the attribute for all the source XDP documents independently to get the same result.

```
<DDX xmlns="http://ns.adobe.com/DDX/1.0/">
<XDP result="result.xdp">
<XDP source="input1.xdp" resolveAssets="all"/>
<XDP source="input2.xdp" resolveAssets="all"/>
<XDP source="input3.xdp" resolveAssets="all"/>
</XDP>
</DDX>
```

Resolve selected source references in an XDP document

You can selectively specify the source references that you want to resolve by specifying the `resolveAssets` attribute for them. The attributes for individual source documents override the resultant XDP document's setting. In this example, the fragments included are also resolved.

```
<DDX xmlns="http://ns.adobe.com/DDX/1.0/">
<XDP result="result.xdp" resolveAssets="all">
<XDP source="input1.xdp" >
<XDPContent source="fragment.xdp" insertionPoint="MyInsertionPoint"
fragment="myFragment"/>
</XDP>
<XDP source="input2.xdp" />
</XDP>
</DDX>
```

Selectively resolve absolute or relative references

You can selectively resolve absolute or relative references in all or some of the source documents, as shown in the example below:

```
<DDX xmlns="http://ns.adobe.com/DDX/1.0/">
<XDP result="result.xdp" resolveAssets="absolute">
<XDP source="input1.xdp" />
<XDP source="input2.xdp" />
</XDP>
</DDX>
```

Dynamically insert form fragments into an XFA form

You can use the Assembler service to create an XFA form that is created from another XFA form that fragments are inserted into. Using this feature, you can use fragments to create multiple forms.

Support for dynamic insertion of form fragments supports single-source control. You maintain a single source of commonly used components. For example, you can create a fragment for your company banner. If the banner changes, you only have to modify the fragment. The other forms that include the fragment are unchanged.

Form designers use Designer to create form fragments. These fragments are uniquely named *subforms* within an XFA form. The form designers also use Designer to create XFA forms that have uniquely named insertion points. You (the programmer) write DDX documents that specify how fragments are inserted into the XFA form.

The following illustration shows two XML forms (XFA templates). The form on the left contains an insertion point named `myInsertionPoint`. The form on the right contains a fragment named `myFragment`.

When the Assembler service interprets the following DDX document, it creates an XML form that contains another XML form. The `myFragment` subform from the `myFragmentSource` document is inserted at the `myInsertionPoint` in the `myFormSource` document.

```
<DDX xmlns="http://ns.adobe.com/DDX/1.0/">
 <XDP result="myFormResult">
   <XDP source="myFormSource">
<XDPContent fragment="myFragment" insertionPoint="myInsertionPoint"
source="myFragmentSource"/>
   </XDP>
 </XDP>
</DDX>
```

Package an XDP document as PDF

You can use the Assembler service to package an XDP document as a PDF document, as shown in this DDX document.

```
<DDX xmlns="http://ns.adobe.com/DDX/1.0/">
 <PDF result="Untitled 1" encryption="passEncProfile1">
```

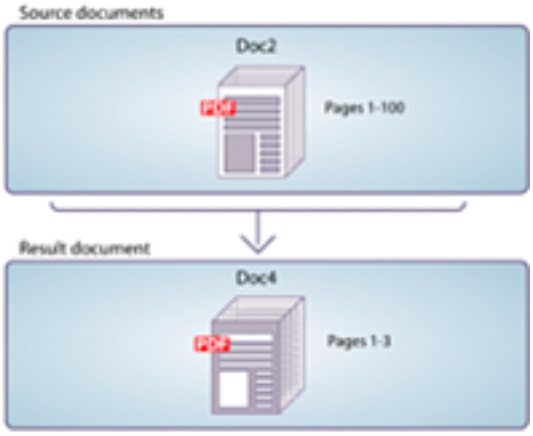
```
    <XDP>
<XDP source="sourceXDP3"/>
<XDP source="sourceXDP4"/>
    </XDP>
  </PDF>
</DDX>
```

Disassemble PDF documents

You can use the Assembler service to disassemble a PDF document. The service can extract pages from the source document or divide a source document based on bookmarks. Typically, this task is useful if the PDF document was originally created from many individual documents, such as a collection of statements.

Extract pages from a source document

In the following illustration, pages 1-3 are extracted from the source document and placed in a new resultant document.

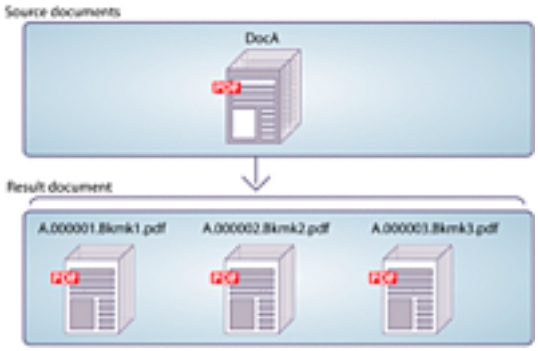


The following example is a DDX document used to disassemble the document.

```
<PDF result="Doc4">
<PDF source="Doc2" pages="1-3"/>
</PDF>
```

Divide a source document based on bookmarks

In the following illustration, DocA is divided into multiple resultant documents. The first level 1 bookmark on a page identifies the start of a new resultant document.



The following example is a DDX document that uses bookmarks to disassemble a source document.

```
<PDFsFromBookmarks prefix="A">
<PDF source="DocA"/>
</PDFsFromBookmarks>
```

Determine whether documents are PDF/A-compliant

You can use the Assembler service to determine whether a PDF document is PDF/A-compliant. *PDF/A* is an archival format meant for long-term preservation of the document's content. The fonts are embedded within the document, and the file is uncompressed. As a result, a PDF/A document is typically larger than a standard PDF document. Also, a PDF/A document does not contain audio and video content.

Obtain information about a PDF document

You can use the Assembler service to obtain the following information about a PDF document:

- Text information.
  - Words on each page of the document
  - Position of each word on each page of the document
  - Sentences in each paragraph of each page of the document
- Bookmarks, including the page number, title, destination, and appearance. You can export this data from a PDF document and import it into a PDF document.
- File attachments, including file information. For page-level attachments, it also includes the location of the file attachment annotation. You can export this data from a PDF document and import it into a PDF document.
- Package files, including file information, folders, package, schema, and field data. You can export this data from a PDF document and import it into a PDF document.

Validate DDX documents

You can use the Assembler service to determine whether a DDX document is valid. For example, if you upgraded from a previous AEM Forms on JEE version, validation ensures that your DDX document is valid.

Call other services

You can use DDX documents that cause the Assembler service to call the following AEM Forms on JEE services. The Assembler service can call only those services installed with AEM Forms on JEE.

Acrobat Reader DC extensions service:

Enables Adobe Reader users to digitally sign the resultant PDF document.

Forms service:

Merges an XDP file and XML data file to produce a PDF document that contains the filled interactive form.

Output service:

Converts a dynamic XML form to a PDF document that contains a non-interactive form (flattens the form). The Assembler service flattens static XML forms and Acrobat forms without calling the Output service. You can specify

DocConverter service:

Converts a PDF document to a PDF/A document.

Generate PDF service:

Converts native file formats to PDF. Examples of native file formats are Word, Excel, and HTML.

Distiller service:

Converts a PostScript document to a PDF document.

DDX documents describe the characteristics of the resultant PDF documents. They do not provide explicit calls to services available in AEM Forms on JEE. When the Assembler service interprets certain DDX elements, it determines whether to call other services to achieve the result specified by the DDX document. For example, when the Assembler service interprets a PDF source file that specifies a non-PDF file, it calls the Generate PDF service to convert that file to PDF. When the Assembler service interprets a PDF source file that contains an XML form (XFA form) and separate XML form data, it calls the Forms service to merge the data into the XML form.

The following DDX example combines two PDF documents and enables Adobe Reader users to digitally sign the resultant PDF document. The `ReaderRights` element inside the `PDF` result element enables the Adobe Reader usage rights.

```
<?xml version="1.0" encoding="UTF-8"?>
<DDX xmlns="http://ns.adobe.com/DDX/1.0/">
<PDF result=”outDoc”>
<PDF source=”doc1”/>
<PDF source=”doc2”/>
<ReaderRights
credentialAlias="LCESCred"
digitalSignatures=”true”/>
```

```
</PDF>
</DDX>
```

Using DDX and the Assembler service to call other AEM Forms on JEE services can simplify your process diagram. It can even reduce the effort you spend customizing your workflows. (See also Assembler Service and DDXReference.)

3.3. Considerations for the Assembler service

Assembling and applying content to large PDF documents can use so much memory that the Assembler service is terminated with an out of memory (OOM) exception. You can use operation checkpoints to avoid triggering such exceptions. See the AssemblerService and DDX Reference.

4. barcoded forms Service

The barcoded forms service extracts data from electronic images of barcodes. The service accepts TIFF and PDF files that include one or more barcodes as input and extracts the barcode data. Barcode data can be formatted in various ways, including XML, delimited string, or any custom format created with JavaScript.

The barcoded forms service supports the following two-dimensional (2D) symbologies supplied as scanned TIFF or PDF documents:

- PDF417
- Data Matrix
- QR Code

The service also supports the following one-dimensional symbologies supplied as scanned TIFF or PDF documents:

- Codabar
- Code128
- Code 3 of 9
- EAN13
- EAN8

IMPORTANT: You must consider the following limitations when using the barcoded forms service:

- The service fully supports AcroForms and static forms containing 2D barcodes that are simply saved using Adobe Reader or Acrobat. However, for 1D barcodes, you must either flatten the form or supply it as scanned PDF or TIFF document.
- Dynamic XFA forms are not fully supported. For the service to properly decode 1D and 2D barcodes in a dynamic form, you must either flatten the form or supply it as scanned PDF or TIFF document.

In addition, barcodes need not originate from forms. The service can decode any barcode that uses supported symbology provided that the above limitations are observed.

For more information on how to create interactive barcoded forms, see Designer Help.

4.1. Using the service

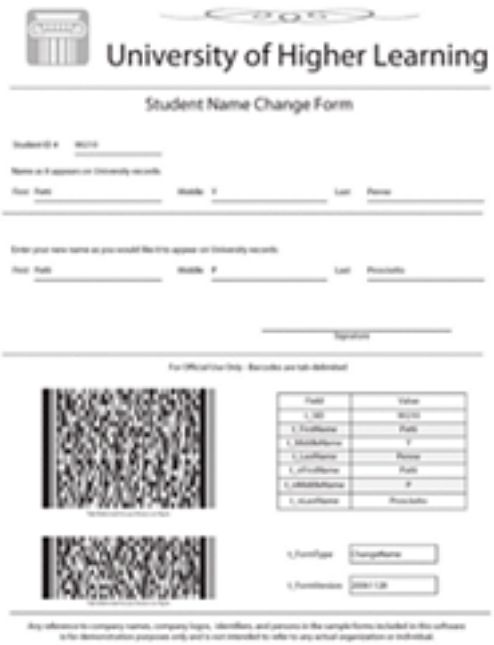
You can accomplish the following tasks by using the Barcoded Forms service:

- Extract barcode data from barcode images (TIFF or PDF). The data is stored as delimited text.
- Convert delimited text data to XML (XDP or XFDF). XML data is easier to parse than delimited text. Also, data in XDP or XFDF format can be used as input for other services in AEM Forms on JEE.

For information about developing processes that use this service, see Workbench Help. For information about developing client applications that programmatically interact with this service, see Programming with AEM Forms.

You can use the Applications and Services pages within the administration console to configure default properties for this service. (See Barcoded Forms service settings in *administration console Help*.)

For each barcode in an image, the barcoded forms service locates the barcode, decodes it, and extracts the data. The service returns the barcode data (using entity encoding where required) in a `content` element of an XML document. For example, the following scanned TIFF image of a form contains two barcodes:



The barcoded forms service returns the following XML document after decoding the barcodes:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xb:scanned_image
xmlns:xb="http://decoder.barcodedforms.adobe.com/xmlbeans" path="tiff"
version="1.0">
<xb:decode>
<xb:date>2007-05-11T15:07:49.965-04:00</xb:date>
<xb:host_name>myhost.adobe.com</xb:host_name>
<xb:status type="success">
<xb:message />
</xb:status>
</xb:decode>
<xb:barcode id="1">
<xb:header symbology="pdf417">
<xb:location page_no="1">
<xb:coordinates>
<xb:point x="0.119526625" y="0.60945123" />
<xb:point x="0.44457594" y="0.60945123" />
<xb:point x="0.44457594" y="0.78445125" />
<xb:point x="0.119526625" y="0.78445125" />
</xb:coordinates>
```

```
</xb:location>
</xb:header>
<xb:body>
<xb:content encoding="utf-8">t_SID t_FirstName t_MiddleName t_LastName
t_nFirstName t_nMiddleName t_nLastName 90210 Patti Y Penne Patti P
Prosciutto</xb:content>
</xb:body>
</xb:barcode>
<xb:barcode id="2">
<xb:header symbology="pdf417">
<xb:location page_no="1">
<xb:coordinates>
<xb:point x="0.119526625" y="0.825" />
<xb:point x="0.44457594" y="0.825" />
<xb:point x="0.44457594" y="0.9167683" />
<xb:point x="0.119526625" y="0.9167683" />
</xb:coordinates>
</xb:location>
 </xb:header>
<xb:body>
<xb:content encoding="utf-8">t_FormType t_FormVersion ChangeName
20061128</xb:content>
 </xb:body>
</xb:barcode>
</xb:scanned_image>
```

4.2. Considerations for the service

Workflows that use barcoded forms

Form authors create interactive barcoded forms by using Designer. (See DesignerHelp.) When a user fills a barcoded form by using Adobe Reader or Acrobat, the barcode is updated automatically to encode the form data.

The barcoded forms service is useful for converting data that exists on paper to electronic format. For example, when a barcoded form is filled and printed, the printed copy can be scanned and used as input to the barcoded forms service.

Watched folder endpoints are typically used to invoke applications that use the barcoded forms service. For example, document scanners can save TIFF or PDF images of barcoded forms in a watched folder. The watched folder endpoint passes the images to the service for decoding.

Recommended encoding and decoding formats

Barcoded form authors are encouraged to use a simple, delimited format (such as tab-delimited) when encoding data in barcodes and to avoid using Carriage Return as the field delimiter. Designer provides a selection of delimited encodings that automatically generate JavaScript script to encode barcodes. The decoded data has the field names on the first line and their values on the second line, with tabs between each field.

When decoding barcodes, specify the character that is used to delimit fields. The character specified for decoding must be the same character that was used for encoding the barcode. For example, when form authors use the recommended tab-delimited format, the Extract to XML operation in processes should use the default value of `Tab` for the field delimiter.

User-specified character sets

When form authors add barcode objects to their forms by using Designer, they can specify a character encoding. The recognized encodings are UTF-8, ISO-8859-1, ISO-8859-2, ISO-8859-7, Shift-JIS, KSC-5601, Big-Five, GB-2312, UTF-16. By default, all data is encoded in barcodes as UTF-8.

When decoding barcodes, you can specify the character set encoding to use. To guarantee that all data is decoded correctly, specify the same character set that the form author specified when the form was designed.

API Limitations

When you use the BCF APIs, certain limitations of the API must be taken into consideration:

- Dynamic forms are not supported.
- Interactive forms are not decoded correctly, unless they are flattened.
- 1-D barcodes should contain only alphanumeric values (if supported). 1-D barcodes containing special symbols are not decoded.

5. Connector Services for ECM

The Connector for EMC Documentum, Connector for IBM Content Manager, Connector for IBM FileNet, and Connector for Microsoft SharePoint provide the following services:

A content repository connector service:

These are independent services usable as operations in a process created in a AEM Forms on JEE process:

- Content Repository Connector for EMC Documentum
- Content Repository Connector for IBM Content Manager
- Content Repository Connector for IBM FileNet
- Content Repository Connector for Microsoft SharePoint

Workbench developers can use these services in a process to store content and retrieve content from custom content models in an ECM content repository. Each connector service provides access to content objects and their metadata stored in the ECM content repository.

If you have installed the Adobe Connector for Microsoft SharePoint, you can enable SharePoint site users to invoke AEM Forms on JEE processes such as converting documents in the SharePoint repository to Adobe PDF format, applying additional usage rights on Adobe PDF documents, and securing documents with Adobe Policy. In addition, SharePoint users can create and initiate workflows that use services available on the connected forms server.

Consider the example of a financial institution automating an account opening process. The process must let applicants digitally sign application forms, archive the completed forms in the ECM repository, retrieve the final declaration and other relevant documentation, assemble those documents into a single PDF file, and deliver the PDF file to applicants through email. The AEM Forms on JEE application that the financial institution develops for this process includes content-repository connector-service operations. These operations are used for the following purposes:

- Storing the completed forms in a customer-defined content object type in the ECM repository.
- Retrieving the content, which can be generated from other ECM applications or assembled from the ECM repository.

Process Engine Connector for IBM FileNet for IBM FileNet service:

Workbench developers can use the Process Engine Connector for IBM FileNet service for the following purposes:

- Creating processes that set and retrieve IBM FileNet Workflow Step parameters
- Dispatching a workflow step in IBM FileNet.

At runtime, assets can be retrieved from the ECM content repository as part of completing a business process. For example, end users can access forms and submit form data from AEM Forms on JEE Workspace, EMC Documentum Webtop, IBM Content Manager client, or IBM FileNet P8 Workplace. Alternatively, a client application can retrieve and store content as part of an automated business process.

NOTE: The Flex Worksapce is deprecated for AEM Forms.

For more information about the content repository connectors services and the Process Engine Connector for IBM FileNet, see Workbench Help.

NOTE: In previous releases of AEM Forms on JEE, assets could be stored in an ECM repository. In newer releases, assets are stored in the native repository, and the repository provider services have been deprecated. The migration of assets from an ECM repository to the AEM Forms on JEE repository is done when you perform an update to AEM Forms on JEE. For more information, see the AEM Forms on JEE Upgrade document for your application server.

5.1. Using the ECM connector services

To access an ECM programmatically using the Repository Service API, administrators must select the appropriate ECM repository service provider. For information about setting up the ECM connector services, see the installation document specific to your application server, from the AEM Forms Documentation page.

When developing a process that uses a repository provider service, you can specify the resource URL to the ECM content repository. The URL Is specified in the properties associated with a service, such as the Forms service. You can also specify the resource URL by using the Forms Service API.

NOTE: Connector for IBM Content Manager does not currently support the repository provider service.

You can also access an ECM content repository by using the Repository Service API to programmatically store and retrieve information. For example, you can obtain a list of files or retrieve specific files stored in an ECM content repository when a file is needed as part of processing an application.

You can use a content repository connector service in a process to interact with content objects in an ECM content repository. When using this service in a process, you can accomplish tasks such as these:

- Access a user-defined content repository (a repository other than the one the repository provider uses)
- Retrieve content and its attributes from the content repository that another service can consume in a subsequent step in the process
- Store content and its attributes in the content repository that another service produced in a previous step in the process
- Get a list of available custom data models from the content repository and map process variables to content attributes in the content repository
- (Connector for Microsoft SharePoint only) Create and initiate SharePoint workflows that use services in AEM Forms on JEE
- (Connector for Microsoft SharePoint only) Convert documents to Adobe PDF format, apply usage rights, and enable additional features in Adobe Reader

The Connector for Microsoft SharePoint provides the following features:

- Allows users to invoke AEM Forms on JEE processes, such as an approval process from within SharePoint
- Allows users to convert documents to Adobe PDF and manage the rights on a file in PDF or native formats

- Provides the ability to create and initiate SharePoint workflows that use services in AEM Forms on JEE
- Enables users to apply usage rights to PDF files to enable additional features in Adobe Reader
- Allows automation of running AEM Forms on JEE processes from within SharePoint workflows
- Enables users to manage assigned AEM Forms on JEE tasks and claim new tasks from within SharePoint.
- Allows integration of AEM Forms on JEE forms with SharePoint Server and effectively use SharePoint as the repository for form data

  For information about developing processes that use these services, see Workbench Help. For information about developing client applications that programmatically interact with these services, see Programming with AEM Forms. For information about configuring SharePoint sites to invoke AEM Forms on JEE processes and perform AEM Forms on JEE actions, see the installation document specific to your application server, from the AEM Forms Documentation page.

  Configure the default connection to the ECM content repository, change the repository service provider, and specify other ECM-specific default settings by using administration console. (For more information, see the Administration Console Help.)

6. (Deprecated) Central Migration Bridge Service

The Central Migration Bridge service is provided for existing Adobe Output Server clients to assist in migrating to AEM Forms on JEE Output. The Adobe Output Server family (Central) includes the following products:

- Adobe Output Designer
- Adobe Central Output Server
- Adobe Central Pro Output Server
- Adobe Web Output Pak

  These products provide features that are similar to Output; however, they are based on more mature Adobe technology.

  The Central Migration Bridge service allows you to use a core set of Central Server functions in the AEM Forms on JEE environment. You will continue using your existing assets (such as MDF output templates and TDF files) with AEM Forms on JEE.

  The Central Migration Bridge service is not intended for general Output customer use. It is targeted at customers that have a Central Output Server implementation and are migrating to AEM Forms on JEE. To use the Central Migration Bridge service, you must have a valid license for Central Pro Output Server 5.7 or an executed Central Pro Output Server 5.7 migration agreement.

6.1. Using the service

The Central Migration Bridge service uses a subset of Adobe Central Pro Output Server functionality. This subset duplicates the functionality provided by the Central Print Agent, Central Transformation Agent, and Central XML Import Agent. With this subset, you can continue using the following Central assets:

- Template design (*.ifd)
- Output templates (*.mdf)
- Data files (*.dat files), including the Central field-nominated format
- Preamble files (*.pre files)
- Data transformation definition files (*.tdf)

  To use the Central Migration Bridge service, you must have proper access rights to the Central installation directory.

  For information about developing processes that use this service, see WorkbenchHelp. For information about developing client applications that programmatically interact with this service, see Programming with AEM Forms.

Central Merge operation

The Central Merge operation merges data with output templates. It returns the result in a format supported by Central, such as IPL, ZPL, PDF, PS, or PCL. This operation can use an external preamble file.

For example, you have a field-nominated data file that contains customer information, such as the name, address, and invoice number. Your application can use the Central Merge operation to merge the data file with an Invoice output template design. Your application can then use the following services available in AEM Forms on JEE to email, save to disk, or print the output file:

- Email service
- File Utilities service
- Output service's sendToPriner operation

  In addition to the primary input parameters (data file and template), this operation accepts a string representing command-line options passed to the Central JFMERGE command. You can use these options to specify all standard options that you use in your current Central job management database (JMD).

  The Central Merge operation can produce a trace file. To enable this feature, specify the trace file option. You can use the trace file to achieve a page 1 of n page numbering scheme on generated documents by performing these tasks:

  a) Run Central Merge once to obtain a trace file.

  b) Pass the resultant trace file to a second run of Central Merge to format the page numbering.

     The Central Merge operation runs the Central JFMERGE command (Central Print Agent). This operation uses the jfmerge.ini file in the same way that Central Pro uses it.

Central Transformation operation

The Central Transformation operation converts structured text and overlay text to acceptable formatted data. The resultant data can then be used with the Central Merge operation. The Central Transformation operation converts data created in ASCII format, including fixed length, character-delimited, overlay, XML, and Central-proprietary field-nominated format. It converts such data to XML or field-nominated format.

This operation also provides a scripting capability that you can use to modify an existing data file. Here are some applications for this scripting capability:

- Modify the data into a format that you can merge with a form design.
- Transform existing data, such as dates, from MM-DD-YYYY to DD-MM-YYYY format.

  In addition to the primary input parameters (data file and TDF file), this operation accepts a string representing command-line options that are passed to the Central JFTRANS command.

  The Central Transformation operation runs the Central JFTRANS command (Central Transformation Agent). This operation uses the jfmerge.ini file in the same way that Central Pro uses it.

Central XML Import operation

The Central XML Import operation transforms well-formed XML data into field-nominated format.

In addition to the primary input parameters (XML file), this operation accepts a string representing command-line options that are passed to the Central XMLIMPORT command.

The Central XML Import operation runs the Central XMLIMPORT command. This operation uses the xmlimport.ini file in the same way that Central Pro uses it.

Central Data Access operation

The Central Data Access operation provides access to certain elements of a field-nominated data file (DAT file) within a AEM Forms on JEE process. It extracts the ^field, ^global, ^form, and ^job commands from the data file and transforms them into an intermediate XML document. Your process can use XPath expressions to access the values of specific entries within that XML file. It can use the accessed values to make decisions, and it can pass the intermediate XML file to other services available in AEM Forms on JEE.

Workbench provides an XPath builder dialog box that simplifies the task of creating XPath expressions.

NOTE: The Central Data Access operation does not fully convert field-nominated data files into an equivalent XML representation. Instead, it converts specific elements into an intermediate XML representation.

The Central Data Access operation makes it easier to integrate existing Central Pro applications into AEM Forms on JEE. Here is an example that uses this operation to determine the values of specific field-nominated commands in the data file:

1) Invoke the Central Data Access operation to obtain the intermediate XML representation of the commands from the data file.
2) Use an XPath expression to get the value of the ^job field-nominated command (the job value).
3) Use the job value to determine which template to use with the data.
4) Invoke the Central Merge operation to merge the selected template with the original data file and to produce a PDF document.
5) Use an XPath expression to get the value of the ^field EMAILADDRESS from the intermediate XML representation.
6) Invoke the Email service to send the PDF file to the job originator through email.

   The Central Data Access operation does not run a Central command.

6.2. Considerations for the service

You can use the Central Migration Bridge service to gradually migrate aspects of your existing Adobe Output Server assets and implementation. AEM Forms on JEE and Central Migration Bridge treat log results differently than Adobe Central Pro Output Server treats them.

Migrating

Migrating libraries of templates and modifying related data sources can be the largest part of any migration effort. Using the Central Migration Bridge service, you can migrate an application from Central to Output in stages.

Initial stage:

Move the invocation and run-time aspects from the Central environment to the AEM Forms environment. You can continue to use existing Central templates and data assets. If you use Central Job Management Database (JMD), replace it with AEM Forms on JEE processes. Output does not support JMD, and it does not provide tools for converting JMD.

If your existing Central templates were saved with a version of Output Designer earlier than 5.7, they may work as is. However, in some instances (especially with pre-5.4 templates), you may need to migrate your existing Central templates by using Output Designer 5.7. When you use that application to open and save a template, it automatically migrates the template. Use Output Designer 5.7 to test migrated templates to ensure that they work as expected.

Subsequent stages:

Over time, convert templates and non-XML data assets for use with Output. You can simultaneously use new and converted processes on the same forms server. Over time you can convert all of your assets and become independent of the Central Migration Bridge service.

You can use Designer to import Output Designer templates. To enable this feature, install Output Designer with Designer. To use this feature, open a file of type Adobe Output Designer template (*.ifd). Designer performs a basic conversion. The degree of completeness of the conversion depends on the complexity of the incoming form. Use Designer to complete the conversion process. (See Designer Help.)

Log results

How you obtain logs that Central produces differs from how you obtain log files that the Central Migration Bridge service produces. Central creates a centralized log file; the Central Migration Bridge service returns the log to the application or process that initiated the operation.

To save the log that the Central Migration Bridge service returned, write the returned log document to a file system. Saving the log is especially useful for debugging.

To adjust the amount of content in the log file, set values in the standard .ini files.

Requirements

The Central Migration Bridge service requires that you install Adobe Central Pro Output Server 5.7 on the same computer that is hosting AEM Forms on JEE. The Central Migration Bridge service does not support earlier versions of Adobe Central Pro Output Server. (See the installation document specific to your application server, from the AEM Forms Help page)

Central Migration Bridge supports the same platforms that AEM Forms on JEE supports.

Samples

AEM Forms on JEE provides samples that demonstrate how to use the Central Migration Bridge service.

7. Convert PDF Service

The Convert PDF service converts PDF documents to PostScript, SWF, or image files (JPEG, JPEG 2000, PNG, and TIFF). Converting a PDF document to PostScript is useful for unattended server-based printing on any PostScript printer. Converting a PDF document to a multipage TIFF file is practical when archiving documents in content management systems that do not support PDF documents.

7.1. Using the Convert PDF service

You can accomplish the following tasks by using the Convert PDF service:

- Convert PDF documents to PostScript. When converting to PostScript, you can use the conversion operation to specify the source document and whether to convert to PostScript level 2 or 3. The PDF document you convert to a PostScript file must be non-interactive.
- Convert PDF documents to JPEG, JPEG 2000, PNG, and TIFF image formats. When converting to any of these image formats, you can use the conversion operation to specify the source document and an image options specification. The specification contains various preferences, such as image conversion format, image resolution, and color conversion.
- Convert PDF documents to SWF. When converting to SWF, you can use the conversion operation to specify the source document and SWF specification.
- (*Deprecated*) Extract files from an archive file to a directory.

  NOTE: The operation to extract files from an archive was deprecated in LiveCycle ES 8.2; however, you can continue using it in AEM Forms on JEE.

For information about developing processes that use this service, see Workbench Help. For information about developing client applications that programmatically interact with this service, see Programming with AEM Forms.

Use the Applications and Services pages of administration console to configure default properties for this service. (See Convert PDF service settings in *AdministrationConsoleHelp*.)

8. Decision Point Service

The Decision Point service is used in Workbench processes for a branching decisions.

Several situations require the use of the Decision Point service:

- Several routes need to be evaluated to determine the first operation to execute in a process.

  This situation occurs when a process is initiated when someone submits a form, and form data determines the first action to execute in the Workbench process. For example, a customer can fill an invoice dispute form through your corporate website. The monetary value claimed by the invoice determines whether the form is routed to a first-level manager for approval or to a credit representative for processing.
- Several different routes in a process converge at a point where a set of rules are evaluated.

  This situation can occur when the process loops to a step where a set of rules are re-evaluated. For example, in a quality assurance process, an issue must go through a retesting process until it is fixed and the process can proceed.

  This situation can also occur if several branches converge after running in parallel. For example, in a process for hiring new employees, when an applicant is hired, several subprocesses are initiated as part of the hiring process. When each of the subprocesses completes, multiple rules based on the data of each subprocess are evaluated to determine the next step.

8.1. Using the Decision Point service

You can use this service in a process when you need multiple routes to originate at an operation, where the branching is not part of another operation in the process. The Decision Point service acts as a node in the process that serves as the origin of many routes but has no executable function itself.

For information about developing processes that use this service, see Workbench Help.

9. Distiller Service

The Distiller® service converts PostScript, Encapsulated PostScript (EPS), and printer text files (PRN) to PDF files. The Distiller service is frequently used to convert large volumes of print documents to electronic documents, such as invoices and statements. Converting documents to PDF also allows enterprises to send their customers a paper version and an electronic version of a document.

NOTE: To convert PostScript files to PDF documents, either Acrobat XI or Microsoft Visual C++ 2005 redistributable package must be installed on the server hosting AEM Forms on JEE.

9.1. Using the Distiller service

When converting a PostScript, Encapsulated PostScript, or PRN file to a PDF file, you can use the conversion operation to specify several options to apply to the resultant PDF document. Here are the parameters you can use to specify these options:

- PDF settings: This parameter provides the name of the Adobe PDF settings to use for the conversion. The named settings are defined on the administration console. The console is pre-configured with several Adobe PDF settings. The names of these settings are locale-specific. On English installations, these names include `High Quality Print`, `PDFA1b 2005 CMYK`, and `Press Quality`.

  If the Input Settings Document parameter provides a value, this parameter is ignored. If this parameter and the Input Settings Document parameter are both null, this operation uses the default file type settings instance that is defined on the forms server.
- Security settings: This parameter provides the name of the security settings to use for the conversion. The named settings are defined on the administration console. On English installations, the console is pre-configured with only the `No Security` security settings.

  If the Input Settings Document parameter provides a value, this parameter is ignored. If this parameter and the Input Settings Document parameter are both null, this operation uses the default file type settings instance that is defined on the AEM Forms on JEE server.
- Input Settings Document: An XML file containing conversion settings, including Adobe PDF settings and security settings. The Input Settings Document can contain multiple sets of settings. This operation uses only the default sets.
- Metadata information to apply to the generated PDF document. Only UTF-8 encoded Adobe Extensible Metadata Platform (XMP) metadata is supported.

For information about developing processes that use this service, see Workbench Help. For information about developing client applications that programmatically interact with this service, see Programming with AEM Forms.

In administration console, you can use the Applications and Services pages to configure default properties of the Distiller service. (See Distiller service settings in *administration console Help*.) You can also use the PDF Generator page to specify default PDF settings and security settings to apply when converting to PDF. (See Configuring PDF Generator in *administration console Help*.)

9.2. Using the PDFG Network Printer to invoke Distiller

You can invoke the Distiller service CreatePDF2 operation by using any of the techniques described in *Howdevelopersinteractwithservices*. You can also submit conversion jobs to that operation by using the PDFG Network Printer (IPP printer).

PDFG Network Printer driver is installed like any other print driver on the desktop. Users can take advantage of the centralized PDF generation that the PDF Generator module provides from any application on their desktop.

Use of the PDFG Network Printer involves the following general steps:

1) Users print documents to the PDFG Network Printer.
2) The PDFG Network Printer converts the submitted file to a PostScript stream and uses the Internet Printing Protocol (IPP) to send the stream to the Distiller service.
3) The Distiller service converts the stream to PDF. The conversion uses the Adobe PDF settings specified by the PDFG Network Printer configuration.
4) The Distiller service sends the conversion results back to the requestor through email. It can also submit the conversion results to another service or process. This feature effectively makes the PDFG Network Printer another endpoint.

Before a user can print to the PDFG Network Printer, an administrator configures the user's system with the client interface for the PDFG Network Printer. For information about installing the print driver, see the *Installing and Deploying AEM Forms on JEE* documents. For information about configuring this feature, see Setting up a PDFG Network Printer in *administration console Help*.

10. DocConverter Service

The DocConverter service transforms signed or unsigned PDF documents, XML forms (typically created using Designer), and Acrobat forms to PDF/A-compliant documents. You can also use this service to validate whether PDF documents are compliant with PDF/A, which is primarily used for archiving.

The DocConverter service is included with the following AEM Forms on JEE modules:

- Output

  To use the DocConverter service with XML forms, you must have Output. To use the DocConverter service with documents that contain signed or unsigned signature fields, you must have Digital Signatures

10.1. Using the DocConverter service

You can accomplish the following tasks by using the DocConverter service:

- *ValidatingwhetherPDFdocumentsarecompliantwithPDF/A*
- *ConvertingPDFdocumentstoPDF/A*

  For information about developing processes that use this service, see Workbench Help. For information about developing client applications that programmatically interact with this service, see Programming with AEM Forms.

Validating whether PDF documents are compliant with PDF/A

Converting PDF documents to PDF/A

You can use the DocConverter service to transform the following types of files to PDF documents that are compliant with PDF/A-1b, PDF/A-2b, and PDF/A-3b standards.

- Signed or unsigned PDF documents
- XML forms (typically created in Designer)
- Acrobat forms
- Associated metadata to aid in understanding the content.
- Schema for any custom metadata.
- Fonts. All fonts are embedded. As a result, a PDF/A document is typically larger than a standard PDF document.

  PDF/A documents cannot contain or use the following items:
- Audio and video content and scripts
- Encryption
- Scripts

The PDF/A standards have restrictions that affect conversion of signatures and forms. When the DocConverter service converts files to PDF/A-1b or PDF/A-2b, it modifies the file to comply with the PDF/A standards. (See Signatures, AdobeXMLforms and Acrobatforms.)

If the DocConverter service cannot completely convert a file to PDF/A, it partially converts the file. In addition to the converted file, the DocConverter service also creates a conversion report. The report indicates fixes that were applied to the file and violations that could not be fixed.

Although PDF/A is the standard for archiving PDF documents, you are not required to use PDF/A for archiving. Standard PDF documents may satisfy your company's requirements. The purpose of the PDF/A standard is to establish a PDF file for long-term archiving and document-preservation needs.

Signatures

By default, the DocConverter service removes all but the last signature from a PDF file. The service captures information about the removed signatures in metadata. It also records the appearance of the removed signatures by adding an equivalent image to the PDF file. The removed signatures cannot be verified.

Users can sign PDF/A documents.

Adobe XML forms

A PDF/A file cannot contain an XML form templates (XFA forms).

The DocConverter service flattens XML form templates that appear in a PDF file, including XML signatures. The service captures information about the removed signatures in metadata. It also records the appearance of the removed signatures into the document by adding an equivalent image to the PDF file. The flattened signatures cannot be verified.

The process of transforming an interactive PDF document to a non-interactive PDF document is called *flattening*. When a PDF document is flattened, form fields retain their graphical appearance but are no longer interactive. Also, the data in the form cannot be extracted by using standard tools; however, the data is still present in the PDF/A.

Acrobat forms

A PDF/A file can contain any type of interactive Acrobat form field, provided those fields are visible. The fields can contain data, but they cannot contain scripts.

11. Email Service

Email is commonly used to distribute content or provide status information as part of an automated process. The Email service enables processes to receive email messages from a POP3 or IMAP server, and send email messages to an SMTP server.

For example, a process uses the Email service to send an email message with a PDF form attachment. The Email service connects to an SMTP server to send the email message with the attachment. The PDF form is designed to let the recipient click Submit after completing the form. The action causes the form to be returned as an attachment to the designated email server. The Email service retrieves the returned email message and stores the completed form in a process data form variable.

When the Email service connects to a POP3 or IMAP server to retrieve an email message, it requires a way to identify a unique message from others in the inbox. Typically, this identification is done by embedding a unique identifier in the subject line, such as the process ID, or by searching for a particular sender. The Email service provides the capability to customize the *from*, *to*, *subject*, and *body* text of an email message. Developers can specify search criteria for a matching email message, such as the sender or subject of the email message.

11.1. Using the Email service

You can interact with the Email service by developing a process in AEM Forms on JEE that uses the service. You can accomplish the following tasks by using the Email service:

- Configure the Email service with default properties for connecting to an SMTP server for sending email messages. Also configure the connection to either a POP3 or IMAP server for receiving messages.
- Receive email messages and attachments from either a POP3 or IMAP email server. You can save metadata about the email, as well as the message content. You can also set filters for email messages, and set properties about the email server and user account to use.
- Send an email message that has one or more attachments to an SMTP server.

  For information about developing processes that use this service, see Workbench Help.

  You can also use the Applications and Services pages of administration console to configure default properties for this service. (See Email service settings in administration console Help.)

12. Encryption Service

The Encryption service enables you to encrypt and decrypt documents. When a document is encrypted, its contents become unreadable. You can encrypt the entire PDF document (including its content, metadata, and attachments), everything other than its metadata, or only the attachments. An authorized user can decrypt the document to obtain access to its contents. If a PDF document is encrypted with a password, the user must specify the open password before the document can be viewed in Adobe Reader or Acrobat. If a PDF document is encrypted with a certificate, the user must decrypt the PDF document with a private key (certificate) he/she owns. The private key used to decrypt the PDF document must correspond to the public key used to encrypt it.

12.1. Using the Encryption service

For information about developing processes that use this service, see Workbench Help. For information about developing client applications that programmatically interact with this service, see Programming with AEM Forms.

You can use the Applications and Services pages of Administration Console to configure default properties for this service. (See Encryption service settings in *administration console Help*.)

Encrypting PDF documents with a password

You can use the Encryption service to encrypt PDF documents with a password. When you encrypt a PDF document with a password, a user must specify the password to open the PDF document in Adobe Reader or Acrobat. You can choose to encrypt the entire PDF document (content, metadata, and attachments), encrypt everything other than its metadata, or encrypt only the attachments. If you encrypt only the document's attachments, users are prompted for a password only when they attempt to access the file attachments.

When encrypting a PDF document with a password, you must specify two separate passwords. One password is used to encrypt and decrypt the PDF document. The other password is used to remove encryption from the PDF document or to modify permissions.

When you use a password to encrypt a PDF document, you can add permissions that specify tasks that the users who receive the document can do. For example, you can specify whether they can sign and fill, edit, or print the PDF document.

A password-encrypted PDF document must be unlocked before another AEM Forms on JEE operation, such as digitally signing the PDF document, can be performed on it. (See *UnlockingencryptedPDFdocuments*.)

NOTE: It is recommended that you do not encrypt a document prior to uploading it to the repository. If you upload an encrypted PDF document to the repository, it cannot decrypt the PDF document and extract the XDP content.

Removing password encryption

You can use the Encryption service to remove password-based encryption from a PDF document. Then users can open the PDF document in Adobe Reader or Acrobat without specifying a password. After password-based encryption is removed from a PDF document, the document is no longer secure.

Encrypting PDF documents with certificates

You can use the Encryption service to encrypt PDF documents with certificates. Certificate-based encryption lets you use public-key cryptography to encrypt documents for specific recipients. Public-key cryptography uses two types of keys:

- A public key, which is stored inside a certificate that can be shared with other users. The public key certificate is in X.509 format and contains a user's public key and identifying information.
- A private key, which you do not share with others.

  Documents are encrypted by using the public keys (certificates) of the users who will receive the document. When users receive an encrypted document, they use their private keys to decrypt it.

  Certificates are typically issued and digitally signed by a certificate authority (CA). A CA is a recognized entity that provides a measure of confidence in the validity of the certificate. Certificates have an expiration date, after which they are no longer valid. In addition, certificate revocation lists (CRLs) provide information about certificates that were revoked prior to their expiration date. Certificate authorities publish CRLs periodically. The revocation status of a certificate can also be retrieved through Online Certificate Status Protocol (OCSP) over the network.

  When you use certificates to encrypt a PDF document, you can add permissions that specify tasks that individual users can do with the document. For example, you can specify whether they can sign and fill, edit, or print the PDF document.

  Before you can encrypt a PDF document with a certificate, you must use Administration Console to add the certificate to AEM Forms on JEE.

  NOTE: It is recommended that you do not encrypt a document prior to uploading it to the repository. If you upload an encrypted PDF document to the repository, it cannot decrypt the PDF document and extract the XDP content.

  A password-encrypted PDF document must be unlocked before another AEM Forms on JEE operation, such as digitally signing the PDF document, can be performed on it. (See Unlockingencrypted-PDFdocuments.)

Removing certificate-based encryption

You can use the Encryption service to remove certificate-based encryption from a PDF document. Then users can open the PDF document in Adobe Reader or Acrobat. To remove encryption from a PDF document that is encrypted with a certificate, you must reference a public key. After encryption is removed from a PDF document, it is no longer secure.

Unlocking encrypted PDF documents

You can use the Encryption service to unlock password-encrypted or certificate-encrypted PDF documents. Attempting to perform a AEM Forms on JEE operation on an encrypted PDF document generates an exception. After you unlock an encrypted PDF document, you can perform one or more operations on it, such as digitally signing it by using the Signature service.

Determining the encryption type

You can use the Encryption service to determine the type of encryption that is protecting a PDF document. Sometimes it is necessary to dynamically determine whether a PDF document is encrypted and, if so, the encryption type. For example, you can determine whether a PDF document is protected with password-based encryption or a Document Security policy. (See *RightsManagementService*.)

A PDF document can be protected by the following encryption types:

- Password-based encryption
- Certificate-based encryption
- A policy that is created by the Document Security service
- Another encryption mechanism

Encrypting XML within a form

You can use the Encryption service to a encrypt XML. You can encrypt the XML to protect sensitive data in XFA forms, by using public key cryptography, for full or partial XDP encryption and decryption.

Encryption

XML Encryption is the process of generating an `<EncryptedData>` tag from the `http://www.w3.org/2001/04/xmlenc namespace`. The input must be XML data. The operation to encrypt XML requires the following inputs:

- **XML**. The XML that must be encrypted must be well-formed. The XML snippet must contain a single root node and the complete XML declaration. If you want to only encrypt a single tag, ensure that it is wrapped in a well-formed XML, before you encrypt the snippet.
- **Recipient Certificates**. The list (collection) of all recipient certificates for which the XML is encrypted. The recipient certificates can point to a `com.adobe.idp.Document`object, a AEM Forms on JEE Truststore alias, or an LDAP URL for the location of the certificate.
- **XPath Expression**. An XPath expression that points to the exact section that you want to encrypt.

  NOTE: *The XPath expression should not point to an attribute, values, or comments, and must point to one element only.*
- **ID attribute**. This unique identifier of the resultant `<EncryptedData>` tag. Each XPath expression needs to be associated with an ID attribute. The association between the recipient, the XPath list, and the ID will be through a container object. An exception is thrown if an ID is not supplied, or if the ID exists in the input document.

- **Enum value**. This value enables the calling application to choose which Symmetric Key Algorithm to use to encrypt XML. The possible values are `AES128`, `AES192`, `AES256`, or `TRIPLEDES`.
- **isXFA property**. A Boolean value that indicates if the content is XFA or not. The default setting for this is `false`. The value of this property, when true, causes the addition of the XFA-specific `EncryptionProperty` property to the XML.

Decryption

When decryption is performed, the original XML section replaces the `<EncryptedData>` tags in an XML file. To decrypt the XML content, this operation requires the private key of the recipient for whom the content has been encrypted. In AEM Forms on JEE, all private keys are stored in the Truststore, so the decryption operation will require the credential alias information, to fetch the correct private key. The operation to encrypt XML requires the following inputs:

- Encrypted XML. The XML content that has been encrypted, in the form of a `com.adobe.idp.Document` object.
- Credential alias. A collection of credential aliases to be used to decrypt the XML. Each credential alias acts as a private key identifier, that points to the right private key to use to decrypt the XML or section of XML.

When you provide a user's credential alias, the decryption operations looks up the corresponding private key from the Truststore, and uses it to decrypt the portions of XML encrypted for that user. However, if you pass `null` in the place of the credential alias, the decryption operation finds all the encrypted portions of the XML, and decrypts all the portions for which it can find a private key in the Truststore.

12.2. Considerations for the Encryption service

Any combination of encrypting, certifying, and applying usage rights to the same document must occur in the following order. These services must be invoked within a short-lived process:

1) Apply encryption (Encryption service) or a policy (Document Security service) to a document before you digitally sign the document (Signature service). A digital signature records the state of the file at the time of signing. Encrypting the document or applying a policy after you apply a signature changes the bytes in the file, causing the signature to appear invalid.
2) Certify a PDF document (Signature service) before you set usage rights (Acrobat Reader DC extensions service). If you certify a document after you apply usage rights, it invalidates the usage rights signature, therefore removing the usage rights from the document.
3) Digitally sign a PDF document (Signature service) after you set usage rights. Signing a PDF document after applying usage rights does not invalidate the usage rights signature.

   Also, you cannot encrypt a PDF document and apply a policy to the same PDF document. Likewise, you cannot apply a policy to an encrypted PDF document.

13. Execute Script Service

The Execute Script service enables you to execute scripts in processes.

13.1. Using the Execute Script service

You can interact with the Execute Script service by developing a process in Workbench that uses the service.

The Execute Script service supports BeanShell 1.3.0, a Java syntax-compatible scripting language for the Java platform. Implicit objects are available to the script. These objects perform the following tasks:

- Provide access to the object manager, process manager, deployment properties, JNDI initial context, and JNDI application context
- Store information that is gathered as a result of executing a script and transfer data to the forms server
- Provide all context data for use during the execution of a script

  For information about developing processes that use this service, see WorkbenchHelp.

  You can use the Applications and Services pages of administration console to configure default properties for this service. (See administration console Help.)

14. FTP Service

The FTP service enables processes to interact with an FTP server. FTP service operations can retrieve files from the FTP server, place files on the FTP server, and delete files from the FTP server. For example, documents such as reports generated from a process may be stored on an FTP server for distribution. Or, an external system may generate some files based on previous steps in a process. In a subsequent step in the process, the files may be transferred to a remote location.

14.1. Using the FTP service

You can interact with the FTP service by developing a process in Workbench that uses the service. You can accomplish the following tasks by using the FTP service:

- Specify the default host, port, and user credentials to connect to the FTP server
- Retrieve a list of files that reside in a directory on an FTP server
- Retrieve multiple files from the FTP server based on a file name pattern
- Retrieve a file from the FTP server and save it to the file system of the forms server
- Retrieve the contents of a file from the FTP server and save the contents as process data
- Upload process data to a directory on the FTP server and save the data as a file
- Upload one or more document values to the FTP server
- Upload a file from the file system of the forms server to a directory on the FTP server
- Delete a file from the FTP server

  For information about developing processes that use this service, see Workbench Help.

  You can use the Applications and Services pages of administration console to configure default properties for this service. (See FTP service settings in administration console Help.)

14.2. Considerations for the FTP service

Consider these factors when developing processes that use this service:

- If you specify local paths to files or directories for operation properties, the paths are interpreted as being on the file system of the forms server.
- The user account used to run the forms server must have the required permissions to interact with the files and file locations that service's operations target.

15. File Utilities Service

The File Utilities service enables processes to interact with the file system of the forms server or other file systems that the server can access.

Files are commonly used to integrate with various systems. A process can use the File Utilities service to read or write files in different formats, such as XML, comma-delimited text, and PDF. A process can also use this service to create a file in a specified directory and set permissions on the file.

The File Utilities service may also be part of a process that dynamically generates documents. For example, a process is scheduled to run every night. This process dynamically generates sales reports in PDF and places them in a directory. The directory is defined based on month and year.

15.1. Using the File Utilities service

You can interact with the File Utilities service by developing a process in Workbench that uses the service. You can accomplish the following tasks by using the File Utilities service:

- Delete a file
- Determine whether a file exists
- Find files and directories in the file system
- Determine whether a path is absolute
- Determine whether a path represents a directory or a file
- Determine whether a path represents a hidden file
- Creates a directory or a directory tree
- Save data as files on the file system
- Read information from files and save it in one of these process data formats:
  - Document
  - String
  - XML document
- Write information from these process data formats to files:
  - Document
  - String
  - XML document
- Manipulate directories and files on the file system

For information about developing processes that use this service, see Workbench Help.

15.2. Considerations for the File Utilities service

Consider these factors when developing processes that use this service:

- If you specify local paths to files or directories for operation properties, the paths are interpreted as being on the file system of the forms server.
- The user account used to run the forms server must have the required permissions to interact with the files and file locations that the service's operations target.

16. Form Augmenter Service

The Form Augmenter service enables a PDF form or Acrobat form to function within Workspace. For example, the Form Augmenter service operations are useful in custom render and submit services for Form, Document Form, or xfaForm variables.

A form that is enabled for Workspace has the following characteristics:

- Buttons will appear hidden when displayed in Workspace. The submission will be invoked on the hidden submit button that was added to the form design as part of the Process Fields form object.
- Submit requests are handled by Workspace, which acts as an intermediary between the forms server and the form.
- Forms can be used both offline and online.

NOTE: The Flex Worksapce is deprecated for AEM Forms.

16.1. Using the Form Augmenter service

You can use the Form Augmenter service operations when you create your custom render and submit services for the Form, Document Form, or xfaForm variables.

With the Form Augmenter service operations, you can perform tasks such as these:

- Enable a PDF form for online use in Workspace. The PDF form must be created in Designer or Acrobat 7.0.5 or later.
- Enable a PDF form for offline use in Workspace. The PDF form must be created in Designer. You cannot specify an Acrobat form.
- Add data fields to the form data, which enables a PDF form created in Designer to be used offline in Workspace. You cannot specify an Acrobat form.
- Retrieve a value from a field on a form. The PDF form must be created within Designer or Acrobat 7.0.5 or later.
- Remove a set of data fields from the form data.

  For information about developing processes that use this service, see Workbench Help.

17. Form Data Integration Service

The Form Data Integration service can import form data into a PDF form and export form data from a PDF form. The import and export operations support two types of PDF forms:

- An Acrobat form (created in Acrobat) is a PDF document that contains form fields.
- An XML form (created in Designer) is a PDF document that conforms to the Adobe XML Forms Architecture (XFA).

Form data can exist in one of the following formats, depending on the type of PDF form:

- An XFDF file, which is an XML version of the Acrobat form data format.
- An XDP file, which is an XML file that contains form field definitions. It can also contain form field data and an embedded PDF file. An XDP file that is generated by Designer can be used only if it carries an embedded base-64-encoded PDF document.

17.1. Using the Form Data Integration service

You can import and export data by using XFDF (Acrobat forms only) or XDP (XML forms only). For example, to import data into a form created in Designer, create a valid XDP XML data source. Consider the following example mortgage application form.

To import data values into this form, you must create an XDP XML data source that corresponds to the form. You cannot use an arbitrary XML data source to import data into a form with the Form Data Integration service. The difference between an arbitrary XML data source and an XDP data source is that an XDP data source conforms to the XML Forms Architecture (XFA). The following XML represents an XDP data source that corresponds to the example mortgage application form.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xfa:datasets xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/">
<xfa:data>
<data>
<Layer>
<closeDate>1/26/2007</closeDate>
<lastName>Johnson</lastName>
<firstName>Jerry</firstName>
```

```
<mailingAddress>JJohnson@NoMailServer.com</mailingAddress>
<city>New York</city>
<zipCode>00501</zipCode>
<state>NY</state>
<dateBirth>26/08/1973</dateBirth>
<middleInitials>D</middleInitials>
<socialSecurityNumber>(555) 555-5555</socialSecurityNumber>
<phoneNumber>5555550000</phoneNumber>
</Layer>
<Mortgage>
<mortgageAmount>295000.00</mortgageAmount>
<monthlyMortgagePayment>1724.54</monthlyMortgagePayment>
<purchasePrice>300000</purchasePrice>
<downPayment>5000</downPayment>
<term>25</term>
<interestRate>5.00</interestRate>
</Mortgage>
</data>
</xfa:data>
</xfa:datasets>
```

For information about developing processes that use this service, see Workbench Help. For information about developing client applications that programmatically interact with this service, see Programming with AEM Forms.

You can use the Applications and Services pages of Administration Console to configure default properties for this service. (See administration console Help.)

Retaining Legacy Appearance of PDF Forms in AEM Forms on JEE

In AEM Forms on JEE, PDF forms rendered with the assistance of Forms, Output, or Form Data Integration (FDI) services have the appearance that is close to PDF forms in Acrobat XI. However, if you are upgrading from the previous version of AEM Forms on JEE, and would like to retain the older appearance of PDF forms, you can set a service-level configuration option for the Forms, Output, and FDI services. For the Forms, Output, or FDI service, when the value of this option (Appearance Compatibility Mode) is set to **9**, all PDF forms rendered using the service appear in the legacy appearance. Any other value set for this option results in generated PDF forms appearing with the look and feel enabled by AEM Forms on JEE.

This option can be set from the administration console (see the ManagingServices topic in the *administration help*) or from Workbench (see Workbench Help).

18. Forms Service

The Forms service enables you to create interactive data capture client applications that validate, process, transform, and deliver forms typically created in Designer. Form authors develop a single form design that the Forms service renders as PDF, HTML, or as Guides (deprecated) in a variety of browser environments that support Adobe Flash Player.

IMPORTANT: Effective March 10, 2012, Adobe is deprecating the Guides capabilities. The Guides functionality is available for upgrade purposes only and will be removed from the product after two major releases.

The Forms service renders interactive PDF forms. An interactive form contains one or more fields for collecting information interactively from a user. An interactive form design produces a form that can be filled online or (in the case of PDF forms) offline. Users can open the form in Acrobat, Adobe Reader, or an HTML browser and enter information into the form's fields. An interactive form can include buttons or commands for common tasks, such as saving data to a file or printing. It can also include drop-down lists, calculations, and validations.

When an end user requests a form, a client application such as a Java servlet sends the request to the Forms service. The Forms service returns the form in an appropriate format to the end user. When the Forms service receives a request for a form, it uses a set of transformations to merge data with a form design. It then delivers the form in a format that best matches the presentation and form-filling capabilities of the target browser. For example, if the end user requests a PDF form, the Forms service renders an interactive PDF form.

The Forms service performs the following functions:

- Provides server-side execution of the intelligence that is in the form design. The Forms service executes the validations and calculations included in the form design and returns the resultant data to the browser.
- Detects whether form design scripts should run on the client or the server. For clients that support client-side scripting such as Internet Explorer 5.0 and later, an appropriate scripting model is loaded into the device so that the scripts can run directly on the client computer.
- Dynamically generates PDF, SWF, or HTML content based the user's preference for a specific form design with or without data. An HTML form can deliver multipage forms page by page. However, a PDF form delivers all the pages at once. In Designer, the form author can script the current page number in the form design. The Forms service can merge one page of data that is submitted at a time or merge only the single page into the form design.
- Supports subforms created in Designer. The Forms service adds extra fields and boilerplate as a result of merging the form design with data or as a result of scripting. With HTML, the added subforms can grow to unlimited page lengths. With PDF, the added subforms paginate at the page lengths that are specified in the form design.
- Renders forms based on fragments. Fragments allow you to share form and script objects that are external to form designs. You can design parts of a form once and reuse them when designing collections of related forms. When creating a new form for the collection, you simply insert a reference to the fragment. When a form author updates a fragment, all forms that contain a reference to the fragment reflect the changes (when the form is rerendered).

- Validates data entry by performing calculations, accessing databases, or enforcing business rules on field-level data.
- Renders forms with file attachments. Also, the Forms service can process form submissions that contain file attachments.
- Displays validation errors in different ways (split frame left, top, right, bottom; no frame left, top, right, bottom; or no UI). This is all done without maintaining any state on the server. The validation errors are also made available in the XML-based validation error document.
- Maintains the state of any pass-through data that the application passed in. *Pass-through data* is data that does not have corresponding fields in the form design being processed. The pass-through data is passed back to the calling application after the target device submits the data.
- Enables a non-technical user to amend a form design by using Designer to meet ongoing business requirements. In contrast, a web application that displays HTML pages may require a user to modify HTML or XML source code to make changes to a web page.

  For information about initially setting up this service, see Configuring AEM Forms on JEE Forms in administration console Help.

18.1. About form types

Before you start working with the Forms service, it is recommended that you understand the various form types that the Forms service uses. This section describes these form types.

Forms that have a flowable layout

A form that has a flowable layout changes based on data prepopulation or through user interaction. A form design that adjusts to accommodate data specifies a set of layout, presentation, and data capture rules. Such a form design includes the ability to calculate values based on user input. The rules are applied when a user enters data into the form or when a server merges data into a form.

A form that has a flowable layout is useful when displaying an undetermined amount of data to users. You do not need to predetermine a fixed layout or number of pages for the form. When a form/form design with a flowable layout is rendered as a PDF form, intelligent page breaks are generated.

Forms that have a fixed layout

A form that have a fixed layout does not change regardless of how much data is placed into the fields. Any fields left unfilled are present in the form but empty. Conversely, if the form contains more data than it can hold, it cannot expand to accommodate the excess data.

Server-side forms

A server-side form can be a data-driven form; that is, the form is populated with data during rendering. The amount of data determines the form’s layout. Multiple data value instances can be provided for a

given field. This causes the field to dynamically replicate so that each data value is displayed within the form.

Fields that are dynamically added to a form are contained in structures called *subforms*, which are located within the form design. An example of a server-side form is one that is part of a custom application that queries a database and retrieves an unknown number of records. After retrieving records from a database, the application merges data into the form. After the data is merged into the form, the application renders the form to a user.

Client-side forms

A client-side form is typically used to collect data from end users by having them click a button (or another control) that produces a new field in which data is entered. The new field appears on the form immediately and does not require a round trip to the server. That is, the form is not sent to AEM Forms on JEE. An example of a client-side form is one that contains fields that where users can enter items to purchase and a button that enables the user to add new fields. Each time the user clicks the button, a new subform is added to the form (a subform can contain a set of related fields).

18.2. How the Forms service processes requests

This section describes how the Forms service processes requests, such as form requests, and specifies the order in which events and scripts are executed.

Requesting a form

When a user requests a form from the Forms service (for example, by clicking a button located on an HTML page), the request initiates a Forms service operation. This table summarizes the interaction among a client device (for example, a web browser), a client application, and the Forms service when a user requests a form.

User actions	Client application actions	Forms service actions
A user requests a form from a web page.	No action	No action
No action	Invokes an operation such as `renderPDFForm`.	No action
No action	No action	Retrieves the form design that is specified.
No action	No action	If data is passed to the Forms service, it prepopulates the form with the data.
No action	No action	Executes all form-wide field initialize events.

User actions	Client application actions	Forms service actions
No action	No action	Executes all form-wide page initialize events.
No action	No action	Executes all form-wide field calculate events.
No action	No action	Executes all form-wide page calculate events.
No action	No action	Executes a page enter event.
No action	No action	Executes a form ready event.
No action	No action	Executes a page enter or exit event.
No action	No action	Transforms the form design into the format specified.
No action	No action	Returns the form to the client application.
No action	Verifies that an error was not returned.	No action
No action	Creates a binary stream and sends it to the client web browser.	No action
Internet Explorer, Mozilla, Netscape Navigator, and Opera browsers perform these actions: • Runs each field initialization marked *Run script on client* *Runs the page initialization marked Run script on client* *Runs each field calculation marked Run script on client* *Runs the page calculations marked Run script on client* **Note:** These actions occur only if the form is rendered as HTML.	No action	No action
Views the form as either PDF or HTML.	No action	No action

Using Form Design buttons

For the Forms service to retrieve form data, perform calculations, or validate field data, the form must provide the mechanism to initiate the request. This initiation is typically accomplished through the use of buttons that are located on the form. The caption displayed on a command button label indicates to the end user the function of the button. When a user clicks a button, the form-related processing is prompted by the script associated with the button. Typically, a button initiates either a submit or a calculate operation.

Buttons are the most common way to initiate logic contained in form design scripts. Placing a button on a form design in Designer and configuring its submit option implies a submit operation. The intent of a submit button is to complete the form and submit data to the Forms service. However, validation operations may interrupt this process. For example, if a user enters an invalid value into a field, they may have to correct the value before the data can be submitted A form can also contain a calculate button. A calculate button can perform calculations on data and update the form.

Submit button

A button can submit form data as XML, PDF, or URL-Encoded (for HTML submissions) data to the Forms service. For example, assume a user fills an interactive form and then clicks a submit button. This action results in the form data being submitted to the Forms service. A client application, such as a Java servlet that is created by using the Forms Service API, can retrieve the data.

A PDF form can submit four types of data (XDP, XML, PDF, and URL-encoded data). An HTML form submits only URL-encoded name-value pairs. By default, when the submission format is PDF, the Forms service captures the PDF data and returns it back out without performing any calculations. You set the submit type in Designer.

The content type of submitted PDF data is `application/pdf`. In contrast, the content type of submitted XML data is `text/xml`. For XDP submissions, it is `application/vnd.adobe.xdp+xml`.

This table summarizes the interaction among a client device (such as a web browser), a client application, and the Forms service when a user clicks a button that initiates a submit operation.

User actions	Client application actions	Forms service actions
A user enters data into form fields and clicks a submit button. This action initiates a submit operation. Client validations marked *run on client* are executed.	No action	No action
Browser performs an HTTP post to the target URL. (This value is defined either in Designer or by the `targetURL` parameter used during the rendition call to the Forms service).	No action	No action

User actions	Client application actions	Forms service actions
No action	Creates a FormServiceClient object, invokes the processFormSubmission method, and passes the HTTP request and headers.	No action
No action	No action	The Forms service merges posted data back into the form (if applicable).
No action	No action	Executes the field click event.
No action	No action	Executes the form-wide field calculate events.
No action	No action	Executes the form-wide page calculate events.
No action	No action	Executes the form-wide field validation events.
No action	No action	Executes the page validation events (which include `validate`, `formatTest`, and`nullTest`).
No action	No action	Executes the Form's `Close` event.
No action	No action	If this validation process fails, it indicates that at least one error exists. The returned processing state value is `Validate`.
No action	Verifies that the Forms service returned a processing state value of `Validate`. In this situation, the result is returned to the client browser so that the user can correct the mistake.	No action

User actions	Client application actions	Forms service actions
For forms that are displayed as HTML, end users see the form that contains the same data, calculations, and list of errors to correct before resubmitting. For Guides (deprecated), end users see the form that contains the same data, calculations, and list of errors to correct before resubmitting. For forms that are displayed as PDF, a user interface is not defined. Validation errors can be retrieved by using the `FormsResult` object's `getValidationErrorsList` method.	No action	No action
No action	No action	If the validation process succeeds, the processing state value is set to `Submit`.
No action	Verifies that the Forms service returns a processing state value of `Submit`. Acknowledges that all form processing is complete. Additional processing is application-specific. For example, a wizard-style application can request the next form panel, do additional data investigations, update the database, or initiate a new workflow process.	No action
The view is application-specific. For example, a new form can be displayed.	No action	No action

Calculate button

A button can be used to execute a calculation operation. When a user clicks a button, the Forms service executes a calculation script located in the form design. The result is rendered back to the web browser with the results displayed in the form. (See *Calculatingformdata*).

This table summarizes the interaction between a client application and the Forms service when a user initiates a calculation operation.

User actions	Client application actions	Forms service actions
A user clicks a button that is located on a form. If the button's `Click` event is marked *run on client*, the form is not submitted to the Forms service. The script is executed in a web browser, Acrobat, or Adobe Reader. A Guide (deprecated) implements an XFASubset in ActionScript, which runs in the Adobe Flash® Player. If the button's `Click` event is marked *run on server*, the form is submitted to the Forms service.	No action	No action
No action	Creates a `FormsServiceClient` object and invokes the `processFormSubmission` method.	No action
No action	No action	The Forms service merges new data into the form design (if applicable).
No action	No action	Executes the field click event.
No action	No action	Executes the form-wide field calculate events.
No action	No action	Executes the form-wide page calculate events.
No action	No action	Executes a page enter or exit event.
No action	No action	Executes the form-wide field validation events.
No action	No action	Executes the page validation event.
No action	No action	Executes the page exit event

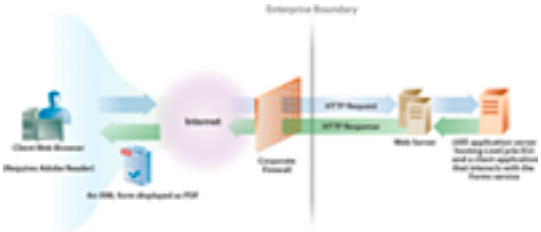
User actions	Client application actions	Forms service actions
No action	No action	Returns the form to the client application that invoked the Forms service. The form's format does not change. If the form is submitted in PDF, it is returned to the client browser in PDF.
No action	Verifies that the Forms service does not return an error.	No action
No action	Creates a binary stream and sends it to the client web browser.	No action
Views calculation results that are displayed in the form.	No action	No action

18.3. Using the Forms service

For information about developing processes that use this service, see Workbench Help. For information about developing client applications that programmatically interact with this service, see Programming with AEM Forms.

Rendering interactive PDF forms

The Forms service renders interactive PDF forms or XDP files to client devices such as a web browser. After an interactive form is rendered, a user can enter data into form fields and click a submit button to send information back to the Forms service. Adobe Reader® or Acrobat must be installed on the computer hosting the client web browser for an interactive PDF form to be visible.



Before you can render a form using the Forms service, you must create a form design. Typically, a form design is created in Designer and is saved as an XDP file.

NOTE: In LiveCycle 7.x releases, the Forms service created non-tagged PDF files by default. In AEM Forms on JEE Services Reference, the Forms service creates tagged PDF files by default.

Rendering forms at the client

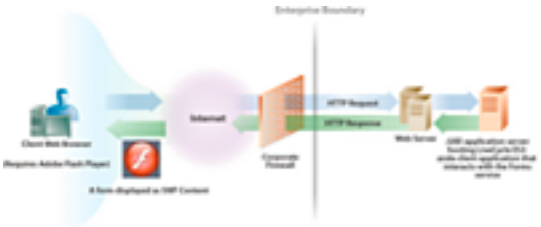
Use the client-side rendering capability of Acrobat or Adobe Reader to optimize the PDF content delivery and improve the Forms service's ability to handle network load This process is known as rendering a form at the client. To render a form at the client, the client device (typically a web browser) must use Acrobat or Acrobat Reader.

Use the client-side rendering capability of Acrobat or Adobe Reader to optimize the PDF content delivery and improve the Forms service's ability to handle network load

Rendering Guides (deprecated)

Guides (deprecated) are based on Adobe Flash® technology and provide a visually engaging and streamlined method for capturing data from a user. Guides (deprecated) can reduce data entry errors through improved usability by logically grouping or simplifying information presented to a user at a given time.

In AEM Forms on JEE, the Forms service can only render legacy form designs created in LiveCycle Designer ES (version 8.x) or later to form Guides (deprecated). The Forms service is not used to render Guides (deprecated), which you create in Workbench using Guide Builder. For information on how to create and render Guides (deprecated).



Rendering forms as HTML

The Forms service renders forms as HTML by retrieving the specified form and transforming it to an HTML form. You can select the type of HTML transformation. Rendering a form as HTML is beneficial. The reason is that the computer that the client web browser is located on does not require Adobe Reader, Acrobat, or Flash Player (for Guides (deprecated)).

To render a form as HTML, the form design must be saved as an XDP file. A form design that is saved as a PDF file cannot be rendered as HTML.

For a form design, it is recommended that the guidelines for layout considerations for HTML forms be followed. This action overcomes text shift issues in a rendered HTML form. (See Using Designer > Creating Forms for Forms > Creating HTML forms > Layout considerations for HTML forms in Designer Help.)

Using Forms IVS to test forms with their intended data sets

The Forms Installation and Verification Sample (Forms IVS) is a sample application for testing your forms with their intended data sets. Forms IVS uses the `renderPDFForm`, `(Deprecated) renderHTMLForm`, and `processFormSubmission` operations that the Forms service exposes.

Forms IVS must be deployed before you can use it. Administrators can use Configuration Manager to deploy Forms IVS. They can also manually deploy it. (See the installation document specific to your application server, from the AEM Forms Documentation page)

To open the Forms IVS application, navigate to http://*[server_name:port_number]*/FormsIVS.

To specify characteristics about your job and to submit your job, click the Test Forms link on the AEM Forms on JEE Forms banner. Here are some of the settings you can specify from the Test Forms window:

- Form to test
- Data file to merge with the form
- Output format
- Performance test selection

  To change settings that Forms IVS uses, click the Preferences link on the AEM Forms on JEE Forms banner. Here are some of the settings you can specify from the Preferences window:
- Options passed to the Forms service
- Locations that Forms IVS obtains form, data, XDC, and companion files from. The locations can be URLs, a repository location, or an absolute reference from a folder on the computer that hosts AEM Forms on JEE. Repository locations can be specified as *repository:/* or *repository:///*.
- Application root and endpoints
- Administrator credentials
- Options for rendering the form

  To view or delete files in the root directory, click the Maintenance link on the Forms banner. You can delete only the files you added to Forms IVS. You cannot delete files that were installed with Forms IVS.

  To see the complete Help for Forms IVS, click the Help link on the Forms banner.

18.4. Forms service options

Retaining Legacy Appearance of PDF Forms

In AEM Forms on JEE, PDF forms rendered with the assistance of Forms, Output, or Form Data Integration (FDI) services have the appearance that is close to PDF forms in Acrobat XI. However, if you are upgrading from the previous version of AEM Forms on JEE, and would like to retain the older appearance of PDF forms, you can set a service-level configuration option for the Forms, Output, and FDI services. For the Forms, Output, or FDI service, when the value of this option (Appearance Compatibility Mode) is set to **9**, all PDF forms rendered using the service appear in the legacy appearance. Any other value set for this option results in generated PDF forms appearing with the look and feel enabled by AEM Forms on JEE.

This option can be set from the administration console (see the Managing Services topic in the *administration help*) or from Workbench (see Workbench Help).

Rendering forms based on fragments

The Forms service can render forms that are based on fragments that you create using Designer. A *fragment* is a reusable part of a form and is saved as a separate XDP file that can be inserted into multiple form designs. For example, a fragment can include an address block or legal text.

Using fragments simplifies and speeds the creation and maintenance of large numbers of forms. When creating a form, insert a reference to the required fragment and the fragment appears in the form. The fragment reference contains a subform that points to the physical XDP file. For information about creating fragments, see Designer Help

A fragment can include several subforms that are wrapped in a choice subform set. *Choice subform sets* control the display of subforms based on the flow of data from a data connection. Use conditional statements to determine which subform from within the set appears in the delivered form. For example, each subform in a set can include information for a particular geographic location. Also, the subform that is displayed can be determined based on the location of the user.

A *script fragment* contains reusable JavaScript functions or values that are stored separately from any particular object, such as a date parser or a web service invocation. These fragments include a single script object that appears as a child of variables in the Hierarchy palette in Designer. Fragments cannot be created from scripts that are properties of other objects, such as event scripts like validate, calculate, or initialize.

Here are advantages of using fragments:

Content reuse:

You can use fragments to reuse content in multiple form designs. To use some of the same content in multiple forms, it is faster and simpler to use a fragment than to copy or re-create the content. Using fragments also ensures that the frequently used parts of a form design have consistent content and appearance in all the referencing forms.

Global updates:

You can use fragments to make global changes to multiple forms only once, in one file. You can change the content, script objects, data bindings, layout, or styles in a fragment. All XDP forms that reference the fragment will reflect the changes.

For example, a common element across many forms may be an address block that includes a drop-down list object for the country. To update the values for the drop-down list object, you must open many forms to make the changes. If you include the address block in a fragment, you only need to open one fragment file to make the changes.

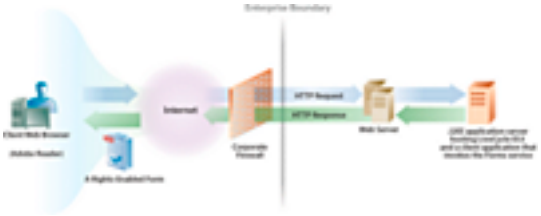
To update a fragment in a PDF form, resave the form in Designer.

Shared form creation:

You can use fragments to share the creation of forms among several resources. Form developers with expertise in scripting or other advanced features of Designer can develop and share fragments that take advantage of scripting and dynamic properties. Form authors can use the fragments to lay out form designs and to ensure that all parts of a form have a consistent appearance and functionality across multiple forms multiple people designed.

Rendering rights-enabled forms

The Forms service can render forms that have usage rights applied to them. Usage rights pertain to functionality that is available by default in Acrobat but not in Adobe Reader, such as the ability to add comments to a form or to fill form fields and save the form. Forms that have usage rights applied to them are called rights-enabled forms. A user who opens a rights-enabled form in Adobe Reader can perform operations that are enabled for that form.



To apply usage rights to a form, the Acrobat Reader DC extensions service must be part of your AEM Forms installation. Also, you must have a valid credential that permits you to apply usage rights to PDF documents. That is, you must properly configure the Acrobat Reader DC extensions service before you can render a rights-enabled form. (See *ReaderExtensionsService*.)

NOTE: To render a form that contains usage rights, use an XDP file as input, not a PDF file. If you use a PDF file as input, the form is still rendered; however, it will not be a rights-enabled form.

NOTE: You cannot prepopulate a form with XML data when you specify the following usage rights: Enable Commenting, Enable Online Commenting, Enable Embedded Attachments, or Enable Digital Signatures.

Handling submitted forms

Web-based applications that enable a user to fill interactive forms require the data to be submitted back to the server. Using the Forms service, you can retrieve the data that the user entered into an interactive form. After you retrieve the data, you can process the data to meet your business requirements. For example, you can perform the following tasks:

- Store the data in a database
- Send the data to another application
- Send the data to another service
- Merge the data in a form design
- Display the data in a web browser

  Form data is submitted to the Forms service as either XML or PDF data. This option is set in Designer. A form that is submitted as XML lets you extract individual field data values. That is, you can extract the value of each form field that the user entered into the form. A form that is submitted as PDF data is binary data, not XML data. As a result, you cannot extract field values. However, you can save the form as a PDF file or send it to another service.

Handling submitted XML data

When form data is submitted as XML, you can retrieve XML data that represents the submitted data. All form fields appear as nodes in an XML schema. The node values correspond to the values that the user

filled in. Consider a loan form where each field in the form appears as a node within the XML data. The value of each node corresponds to the value that a user fills in. Assume a user fills the loan form with data shown in the following form.

The following illustration shows corresponding XML data that is retrieved by using the Forms service.

The form design must be configured correctly in Designer in order for data to be submitted as XML data. To properly configure it to submit XML data, ensure that the Submit button on the form design is set to submit XML data. (See Designer Help.)

Also, you must specify the correct content type to handle XML data. For example, specify `application/vnd.adobe.xdp+xml`. The content type must match the submitted form data. You can also specify multiple content type values, such as the following value, to ensure that various form submissions can be used:

```
CONTENT_TYPE=application/pdf&CONTENT_TYPE=application/vnd.adobe.xdp+xml
```

Handling submitted PDF data

Consider a web application that invokes the Forms service. After the Forms service renders an interactive PDF form to a client browser, the user fills an interactive form and submits it back as PDF data. When the Forms service receives the PDF data, the service can send it to another service or save it as a PDF file. To handle a submitted PDF form, ensure that you specify `CONTENT_TYPE=application/pdf` as the content type.

Prepopulating forms

Prepopulating forms displays data to users within a rendered form. For example, assume a user logs in to a website with a user name and password. If authentication is successful, the custom application queries a database for user information. The data is merged into the form and the form is then rendered to the user. As a result, the user can view personalized data within the form.

Prepopulating a form has the following advantages:

- Enables the user to view custom data in a form
- Reduces the amount of typing the user does to fill a form
- Ensures data integrity by having control over where data is placed

The following two XML data sources can prepopulate a form:

- An XDP data source, which is XML that conforms to XFA syntax (or XFDF data to prepopulate a form created using Acrobat).
- An arbitrary XML data source that contains name/value pairs matching the form's field names.

An XML element must exist for every form field you want to prepopulate. The XML element name must match the field name. An XML element is ignored if it does not correspond to a form field or if the XML element name does not match the field name. It is not necessary to match the order that the XML elements are displayed in if all XML elements are specified.

When you prepopulate a form that already contains data, specify the data that is already displayed within the XML data source. Assume that a form containing 10 fields has data in 4 fields. Next, assume that you want to prepopulate the remaining 6 fields. In this situation, you must specify 10 XML elements in the XML data source that is used to prepopulate the form. If you specify only 6 elements, the original 4 fields are empty.

For example, to prepopulate a confirmation form, you must create an XML data source containing three XML elements that match the three form fields. This form contains the following three fields: `FirstName`, `LastName`, *and* `Amount`. *The first step is to create an XML data source that contains XML elements that match the fields in the form design. This step is shown in the following XML code.*

```
<Untitled>
<FirstName>
<LastName>
<Amount>
</Untitled>
```

The next step is to assign data values to the XML elements, as shown in the following XML code.

```
<Untitled>
<FirstName>Jerry</FirstName>
<LastName>Johnson</LastName>
<Amount>250000</Amount>
</Untitled>
```

After you prepopulate the confirmation form with this XML data source and render the form, data values that you assigned to the XML elements are displayed. The displayed data values are shown in this illustration.

Prepopulating forms with a flowable layout

Forms with a flowable layout are useful to display an undetermined amount of data to users. The layout of the form adjusts automatically to the amount of data that is merged. Therefore, predetermining a number of pages is not required as with a form with a fixed layout.

A form with a flowable layout is typically populated with data that is obtained during run time. As a result, you can prepopulate a form by creating an in-memory XML data source and placing the data directly into the data source.

The following illustration shows an example of a purchase order form with a flowable layout.

A.

Represents the dynamic portion of the form.

B.

Represents the form's header data.

NOTE: Forms can be prepopulated with data from other sources, such as an enterprise database or external applications.

Understanding data subgroups

An XML data source is used to prepopulate a form. An XML data source that is used to prepopulate a form with a flowable layout contains repeating data subgroups. The following XML code shows the XML data source used to prepopulate the purchase order form.

```
<header>
<!-- XML elements used to prepopulate non-repeating fields such as address
<!and city
<txtPONum>8745236985</txtPONum>
<dtmDate>2004-02-08</dtmDate>
<txtOrderedByCompanyName>Any Company Name</txtOrderedByCompanyName>
<txtOrderedByAddress>555, Any Blvd.</txtOrderedByAddress>
<txtOrderedByCity>Any City</txtOrderedByCity>
<txtOrderedByStateProv>ST</txtOrderedByStateProv>
<txtOrderedByZipCode>12345</txtOrderedByZipCode>
<txtOrderedByCountry>Any Country</txtOrderedByCountry>
<txtOrderedByPhone>(123) 456-7890</txtOrderedByPhone>
<txtOrderedByFax>(123) 456-7899</txtOrderedByFax>
<txtOrderedByContactName>Contact Name</txtOrderedByContactName>
<txtDeliverToCompanyName>Any Company Name</txtDeliverToCompanyName>
<txtDeliverToAddress>7895, Any Street</txtDeliverToAddress>
<txtDeliverToCity>Any City</txtDeliverToCity>
<txtDeliverToStateProv>ST</txtDeliverToStateProv>
<txtDeliverToZipCode>12346</txtDeliverToZipCode>
<txtDeliverToCountry>Any Country</txtDeliverToCountry>
<txtDeliverToPhone>(123) 456-7891</txtDeliverToPhone>
<txtDeliverToFax>(123) 456-7899</txtDeliverToFax>
<txtDeliverToContactName>Contact Name</txtDeliverToContactName>
</header>
<detail>
<!-- A data subgroup that contains information about the monitor>
<txtPartNum>00010-100</txtPartNum>
<txtDescription>Monitor</txtDescription>
<numQty>1</numQty>
<numUnitPrice>350.00</numUnitPrice>
</detail>
<detail>
<!-- A data subgroup that contains information about the desk lamp>
<txtPartNum>00010-200</txtPartNum>
<txtDescription>Desk lamps</txtDescription>
<numQty>3</numQty>
<numUnitPrice>55.00</numUnitPrice>
</detail>
<detail>
<!-- A data subgroup that contains information about the Phone>
<txtPartNum>00025-275</txtPartNum>
<txtDescription>Phone</txtDescription>
<numQty>5</numQty>
<numUnitPrice>85.00</numUnitPrice>
</detail>
<detail>
<!-- A data subgroup that contains information about the address book>
<txtPartNum>00300-896</txtPartNum>
<txtDescription>Address book</txtDescription>
<numQty>2</numQty>
```

```
<numUnitPrice>15.00</numUnitPrice>
</detail>
```

Notice that each data subgroup contains four XML elements that correspond to this information:

- Items part number
- Items description
- Quantity of items
- Unit price

  The name of a data subgroup's parent XML element must match the name of the subform in the form design. For example, in the previous illustration, notice that the name of the data subgroup's parent XML element is `detail`*. This corresponds to the name of the subform located in the form design that the purchase order form is based on. If the name of the data subgroup's parent XML element and the subform do not match, a server-side form is not prepopulated.*

  Each data subgroup must contain XML elements that match the field names in the subform. The `detail` *subform in the form design contains the following fields:*

- txtPartNum
- txtDescription
- numQty
- numUnitPrice

Calculating form data

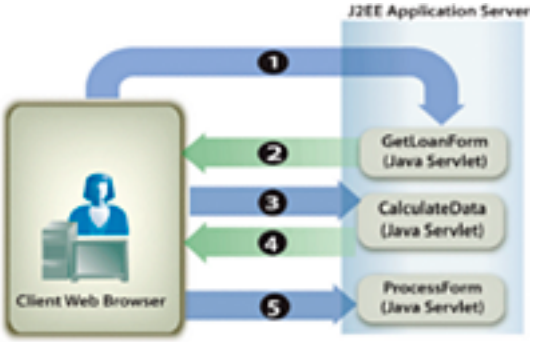
The Forms service can calculate the values that a user enters into a form and display the results. To calculate form data, create a form design script that calculates form data. A form design supports three types of scripts. One script type runs on the client, another runs on the server, and the third runs on both the server and the client. The script type discussed in this topic runs on the server. Server-side calculations are supported for HTML, PDF, and Guide (deprecated) transformations.

As part of the form design process, you can make use of calculations and scripts to provide a richer user experience. Calculations and scripts can be added to most form fields and objects.

The user enters values into the form and clicks the Calculate button to view the results. The following process describes an example application that lets a user calculate data:

- The user accesses an HTML page named *StartLoan* that acts as the web application's start page. This page invokes a Java Servlet named `GetLoanForm`.
- The `GetLoanForm` servlet renders a loan form. This form contains a script, interactive fields, a calculate button, and a submit button.
- The user enters values into the form fields and clicks the Calculate button. The form is sent to the `CalculateData` Java Servlet where the script is executed. The form is returned to the user with the calculation results displayed in the form.
- The user continues entering and calculating values until a satisfactory result is displayed. When satisfied, the user clicks the Submit button to process the form. The form is sent to another Java Servlet named `ProcessForm` that is responsible for retrieving submitted data. (See *Handlingsubmittedforms*.)

The following illustration shows the application's logic flow.



The steps in the diagram are as follows:

a) The `GetLoanForm` Java™ Servlet is invoked from the HTML start page.

b) The `GetLoanForm` Java Servlet uses the Forms service Client API to render the loan form to the client web browser. Rendering a form containing a script that is configured to run on the server differs from rendering a form that does not contain a script. The difference is that you must specify the target location that is used to execute the script. If a target location is not specified, a script that is configured to run on the server is not executed. For example, consider the application introduced in this section. The `CalculateData` Java Servlet is the target location where the script is executed.

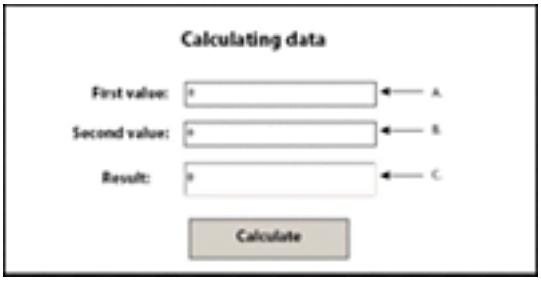
c) The user enters data into interactive fields and clicks the Calculate button. The form is sent to the `CalculateData` Java Servlet, where the script is executed.

d) The form is rendered back to the web browser with the calculation results displayed in the form.

e) The user clicks the Submit button when the values are satisfactory. The form is sent to another Java Servlet named `ProcessForm`.

Typically, a form that is submitted as PDF content contains scripts that are executed on the client. However, server-side calculations can also be executed. A Submit button cannot be used to calculate scripts. In this situation, calculations are not executed because the Forms service considers the interaction to be complete.

To illustrate the usage of a form design script, this section examines a simple interactive form that contains a script that is configured to run on the server. The following illustration shows a form design containing a script. The script adds values that a user enters into the first two fields and displays the result in the third field.



A.

A field named NumericField1

B.

A field named NumericField2

C.

A field named NumericField3

The syntax of the script in this form design is as follows:

```
NumericField3 = NumericField2 + NumericField1
```

In this form design, the Calculate button is a command button, and the script is located in this button's `Click` event. When a user enters values into the first two fields and clicks the Calculate button, the script is executed. The Forms service renders the form back to the client device with the results of the calculation displayed in the NumericField3 field.

For information about creating a form design script, see Designer Help.

Creating Bookmarks

The Forms service enables you to create bookmarks in an XFA form, similar to how it is done in an PDF documents. A bookmark can be defined at any XFA container like a subform, field, or an area on a subform.

To create a bookmark, use the `<extras>` tag in any valid container. A container can have multiple bookmarks. However, only one `<extras>` tag is allowed in each container. To specify more than one bookmark, wrap more `<extras>` tags within the first `<extras>` tag. Also, AEM Forms on JEE supports up to two levels of nesting for <extras> tags.

A following code snippet is a sample of how a bookmark can be inserted in the XFA form:

```
<subform> <!-- Any container element like subform, field or area-->
<extras name="bookmark">
<text name="name">ANY_NAME</text>
<text name="color">0,0,0|R,G,B</text>
<text name="style">normal | italic | bold | bold-italic</text>
<text name="action">gotoPage | setFocus | runScript</text>
<text name="script">ANY_FORM_LEVEL_SCRIPT</text> <!-- JavaScript only. No
formcalc support -->
<extras>
...
</subform>
```

The parameters that must be used to describe the bookmark are:

Parameter	Description
name	The name of the bookmark that will appear in the bookmark pane. If it is not specified, the bookmark will not be generated.

Parameter	Description
color	The color in which the bookmark name is rendered. The color parameter should be indicated in the RGB scheme. For example, to insert a bookmark in red color, this parameter should be specified as 255,0,0. The default value for the color parameter is 0,0,0 (black).
style	The style in which the bookmark name is rendered. The default value of the style parameter is none. Other values can be `bold`, `italic`, or `bold-italic`.
action	The action that is performed when the bookmark is clicked. Values can be: • *`gotoPage`*: This is the default value. Focus is shifted to the page where the parent subform starts. *`setFocus`: Can be used when the parent container is a field. Sets the parent field in focus.* *`runScript`: Triggers JavaScripts to be run (This value is ignored in PDF/A documents).*
script	Relevant, when the value of the action parameter is set to runScript. Supports only JavaScript, that contains script objects present in the document scope.

Generate the document to view the bookmarks. Bookmarks authored this way are available in static and dynamic documents. Bookmarks are compatible with Acrobat Reader 9 and Acrobat Reader X. However, bookmarks can be added to any XFA version document.

Current limitations:

- To ensure compatibility with existing Reader versions, some capabilities are limited.
- For security reasons, a user can change the value of any bookmark parameter, except action and script.
- For dynamic forms, you cannot make changes using the script. AEM Forms on JEE will always generate bookmarks based on what the user has specified in the template. For example, assume that a form contains a select-one-subform type subformset, and the author has defined bookmarks for all the subforms of the subformset. In such a scenario, for dynamic PDF documents, AEM Forms on JEE will generate bookmarks for all subforms, and for static PDF documents, AEM Forms on JEE will generate one bookmark, depending on which subform gets selected based on user input.
- Bookmarks defined for XFA containers that form master page content are ignored.
- Bookmarks specified on containers other than subform, area, field, and draw objects are not supported, and may result in unpredictable behavior.
- If the rendered PDF is archived, bookmarks with value `runScript` for the action parameter are disabled.
- If an interactive form with bookmarks is flattened using the Output service, it will not retain the bookmarks inserted after the generation of PDF using Acrobat or Assembler.

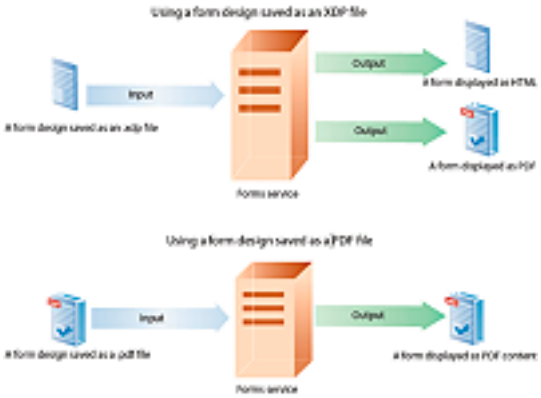
18.5. Considerations for the Forms service

When working with the Forms service, certain considerations are related to the following topics:

- Planning your form design
- Creating form designs for the Forms service
- Rendering forms that contain images
- Rendering forms that include secured images
- Changes to image fields using a script are not retained
- Ensuring fonts are available to AEM Forms on JEE

Planning your form design

Creating application logic by using the Forms service is only one aspect of creating a client application. The Forms service requires form designs that are typically created in Designer (forms can be created in Acrobat as well). Form designs are XML templates that are saved as either XDP files or PDF files. Depending on the type of input, the Forms service outputs forms that are displayed as PDF or HTML forms. The following illustration shows the valid input and output of the Forms service.



The first step in planning your application is to determine the output format of the forms. Save the form design in one of the following ways:

- As an XDP file to have the Forms service output a form displayed as PDF or HTML.
- As a PDF or XDP to have the Forms service output PDF only.

Creating form designs for the Forms service

Behavioral differences exist between form designs that are used to render PDF and HTML. Form designs rendered as PDF are viewed using Acrobat or Adobe Reader.

If you are rendering a form as HTML, some client devices (for example, older browsers) do not provide the same support level for individual object properties. To create a single form design that reduces these limitations, follow this process:

1) To determine how objects behave in a particular client device, see the Transformation Reference.
2) If you are designing a form with a fixed layout and want to output the form as HTML, enable transformation caching. (See Designer Help.)
3) When creating a form design, try to work around limitations by finding ways to implement the form without relying on unsupported object properties.
4) If required, include a layout that works for both PDF and HTML.
5) Read "Creating Accessible Forms" in Designer Help and use the guidelines to build accessibility into your form design.
6) Ask your form developer where scripts should run. By default, scripts run on the client. If the scripts you include in a form design must run on the server, or both the client and server, you may have to change the default setting. For example, a form design may contain a script that extracts data from an enterprise database that is available only on the server. In this situation, the default setting must be modified so that the script runs at the server.
7) Periodically preview the form by using Designer or the client device (for example, a web browser) to troubleshoot problems early in the design process.
8) If the Forms service will be prepopulating forms with data, use test data to thoroughly test your form design.

   For the Forms service to retrieve form data, perform calculations, or validate field data, the form must provide the mechanism to initiate the request. This is typically accomplished through the use of buttons located on the form design. The caption that is displayed on a command button label indicates the function of the button to the end user. When a user clicks a button, the form-related processing is prompted by the script that is associated with the button. Typically, a button initiates either a submit operation or a calculate operation.

   Buttons are the most common way to initiate logic that is contained in form design scripts. Placing a button on a form design in Designer and configuring its submit option implies a submit operation. The intent of a submit button is to complete the form and submit data to the Forms service. However, validation operations may interrupt this process. For example, if a user enters an incorrect value, the user may have to correct the value before the form data can be submitted. Placing other button types on the form implies a calculate operation. The intent of a calculate operation is to run calculations and update the form before a submit operation.

Rendering forms that contain images

When rendering forms with data that contains references to images, the images may not be displayed properly in the rendered form. To ensure that the images are rendered successfully when rendering the form on the client side, set the following run-time options:

- PDFVersion=1.6 (or higher)
- CacheEnabled=0
- renderAtClient=Auto

  To render the form with data on the server side, set the following run-time option: CacheEnabled=0.

Rendering a PDF form that includes a secured image

You may encounter an exception when rendering an HTML form that contains an image that is secured by a certificate. For example, if you use the Forms IVS application to render a form that contains a secured image, an exception may occur.

To resolve this issue, ensure that the J2EE application server is started with the same operating system user name and password values used to add the client side. After you perform this task, the Forms service can access the credential that is required to obtain the image.

Changes to image fields using a script are not retained

Changes made to an image field located on a form design by using client-side script, such as JavaScript, are not submitted and cannot be obtained by a client application. For example, changes made to an image field by a script, which is then followed by a next or previous page action does not retain the change made by the script. This is applicable to forms that are rendered as HTML.

Enabling debug options

You can enable debug options when rendering a PDF form, rights-enabled PDF form, or Guide (deprecated). When enabling the debug option, you can obtain additional information, such as the value of run-time values. To set debug options for the `renderPDFForm` or `renderPDFFormWithUsageRights` operations within in a process created in Workbench, create a variable of type `PDFFormRenderSpec`. Then, using the `setValue` operation, set `/process_data/pdfFormRenderSpecVar/object/@debugEnabled` to `true`. Similarly, for the `(Deprecated) renderHTMLForm` operation, use a variable of type `HTMLRenderSpec`.

Verifying fonts are available to AEM Forms on JEE

Ensure that fonts that are available within a form are available for use on the server hosting AEM Forms on JEE. For example, consider the following scenario. A form author adds a font to the font directory that Designer uses and creates a form on a separate computer. For the Forms service to use that font, ensure that the font is deployed on the J2EE application server that AEM Forms on JEE is installed on. Otherwise, the font cannot be used. You can use Administration Console to deploy fonts.

NOTE: For information about setting up fonts for the Output service, see Specifying fonts to embed in *administration console Help*.

Nested E4X Operators

Scripts using nested E4X double-dot operators do not work correctly. It may result in an error message in the log file similar to:

```
Error: .. cannot work with instances of this class
```

Consider the following script:

```
var person = <person><name>Bob
Smith</name><likes><os>Linux</os><browser>Firefox</browser><language>Jav
```

```
aScript</language><language>Python</language></likes></person>;
alert(person..likes); //This line of code executes correctly
alert(person..likes..os); //This line of code fails to execute
```

The line of code that will fail is:

```
alert(person..likes..os)
```

A workaround for this error is to replace the line of code with:

```
var likes = person..likes;
alert(likes..os);
```

19. Generate PDF Service

The Generate PDF service converts native file formats to PDF. It also converts PDF to other file formats and optimizes the size of PDF documents.

The Generate PDF service uses native applications to convert the following file formats to PDF. Unless otherwise indicated, only the German, French, English, and Japanese versions of these applications are supported. *Windows only* indicates support for only Windows Server 2008.

19.1. Supported Formats for Conversion to PDF

- *Adobe Acrobat DC Latest:* XPS, image formats (BMP, GIF, JPEG, JPG, TIF, TIFF, PNG, JPF, JPX, JP2, J2K, J2C, JPC), HTML, HTM, DWG, DXF, and DWF
- *Adobe Acrobat DC Pro:* XPS, image formats (BMP, GIF, JPEG, JPG, TIF, TIFF, PNG, JPF, JPX, JP2, J2K, J2C, JPC), HTML, HTM, DWG, DXF, and DWF
- *Microsoft Office 2016:* DOC, DOCX, XLS, XLSX, PPT, PPTX, RTF, and TXT
- *Microsoft Office 2013:* DOC, DOCX, XLS, XLSX, PPT, PPTX, RTF, and TXT
- *WordPerfect X7:* WP, WPD
- *Microsoft Office Visio 2016:* VSD, VSDX
- *Microsoft Office Visio 2013:* VSD, VSDX
- *Microsoft Publisher 2016:* PUB
- *Microsoft Publisher 2013:* PUB
- *Microsoft Project 2016:* MPP
- *Microsoft Project 2013:* MPP
- *OpenOffice 4.1.2:* ODT, ODP, ODS, ODG, ODF, SXW, SXI, SXC, SXD, XLS, XLSX, DOC, DOCX, PPT, PPTX,image formats (BMP, GIF, JPEG, JPG, TIF, TIFF, PNG, JPF, JPX, JP2, J2K, J2C, JPC), HTML, HTM, RTF, and TXT
- *OpenOffice 3.4:* ODT, ODP, ODS, ODG, ODF, SXW, SXI, SXC, SXD, XLS, XLSX, DOC, DOCX, PPT, PPTX,image formats (BMP, GIF, JPEG, JPG, TIF, TIFF, PNG, JPF, JPX, JP2, J2K, J2C, JPC), HTML, HTM, RTF, and TXT

NOTE:

PDF Generator supports only English, French, German, and Japanese versions of the supported operating systems and applications.

In addition:

- PDF Generator is not supported on Windows 2008 R1. PDF Generator on Windows requires Windows 2008 R2 SP1 and later.
- PDF Generator requires 32-bit version of Acrobat DC to perform the conversion.
- PDF Generator does not support Microsoft Office 365.

- PDF Generator supports only the 32-bit Retail version of Microsoft Office Professional Plus and other software required for conversion.
- The HTML2PDF service is deprecated on AIX.
- PDF Generator conversions for OpenOffice are supported only on Windows, Linux, and Solaris.
- The OCR PDF, Optimize PDF, and Export PDF features are supported only on Windows.
- Acrobat DC Pro installer is bundled with AEM Forms to enable PDF Generator functionality. Acrobat DC Pro should only be accessed programmatically by AEM Forms, during the term of the AEM Forms license, for use with AEM Forms PDF Generator. For more information, refer to AEM Forms product description as per your deployment "On-Premise or Managed Services".

The Generate PDF service converts the following standards-based file formats to PDF.

- Video formats: SWF, FLV (Windows only)
- Image formats: JPEG, JPG, JP2, J2Kí, JPC, J2C, GIF, BMP, TIFF, TIF, PNG, JPF
- HTML

NOTE: AEM Forms on JEE supports only 32-bit editions of the above mentioned software.

The Generate PDF service converts PDF to the following file formats (Windows only):

- Encapsulated PostScript (EPS)
- HTML 3.2
- HTML 4.01 with CSS 1.0
- DOC (Microsoft Word format)
- RTF
- Text (both accessible and plain)
- XML
- PDF/A-1b and PDF/A-1a that use only the DeviceRGB color space
- PDF/E-1 that uses only the DeviceRGB color space
- DOCX
- XLSX
- PPTX

NOTE: The PDF conversion of FM, PMD, PM6, P65, PM, DWG, MPP, SWF, XPS, FLV, VSD, WordPerfect documents, and Microsoft Office documents (DOC, XLS, PPT, WPD, MPP, RTF, TXT) is possible only if Acrobat XI Pro is installed.

The Generate PDF service requires that you perform these administrative tasks:

- Install required native applications on the computer hosting AEM Forms on JEE
- Install Adobe Acrobat DC Pro on the computer hosting AEM Forms on JEE
- Perform post-installation setup tasks

These tasks are described in Installing and Deploying LiveCycle Using JBossTurnkey.

20. JDBC Service

The JDBC service enables processes to interact with databases. For example, a process may require that data submitted from a form populate an internal database or that data from a database prepopulate a form. The process can contain application logic that binds data from the form fields to the underlying database. Fields may be mapped to database columns. Separate line items in a filled form may also create multiple rows in the database. If a stored procedure that populates the database already exists, a process can use the stored procedure to update the data.

A process may also execute a business rule that is dependent on data stored in a database. For example, if a customer is behind on payment, the order is not shipped, and the customer is sent an email reminder. A process can query the database and look up the status based on the customer ID and create a rule based on this value.

20.1. Using the JDBC service

You can interact with the JDBC service by developing a process in Workbench that uses the service. You can accomplish the following tasks by using this service:

- Specify the data source to use to connect to the database server
- Execute a stored procedure on the database
- Execute an SQL statement on a database server and return the number of rows that were affected
- Query the database using an SQL statement and return the result set as XML data
- Query the database using an SQL statement and save the first row of the result set

  For information about developing processes that use this service, see Workbench Help.

  You can use the Applications and Services pages in administration console to configure default properties for this service. (See JDBC service settings in administration console Help.)

20.2. Considerations for the JDBC service

The data source that is used to connect to the data server must be defined on the application server that hosts the AEM Forms on JEE server. The default value is the JNDI name of the data source for the AEM Forms on JEE database.

To call stored procedures or execute SQL statements, the database user account that is used to access the database must have the required database permissions.

21. LDAP Service

The LDAP service provides operations for querying LDAP directories. LDAP directories are generally used to store information about the people, groups, and services in an organization.

For example, LDAP directories typically store information about the business unit that a person belongs to, information that identifies the person, and information about how to contact them, such as telephone numbers and email addresses. A process can use the LDAP service to query the details based on the user's ID, and map the details to a process variable in order to populate a form.

21.1. Using the LDAP service

You can interact with the LDAP service by developing a process in Workbench that uses the service. You can accomplish the following tasks by using this service:

- Perform a search on the LDAP server and return the results, which you can save as process data.
- Perform a search on the LDAP server and return the results in an XML document, which you can save as process data.

  For information about developing processes that use this service, see Workbench Help.

  You can use the Applications and Services pages in administration console to configure default properties for this service. (See LDAP service settings in administration console Help.)

21.2. Considerations for the LDAP service

Consider these factors when developing processes that use this service:

- You must configure the properties that are used to connect to the LDAP server before using the LDAP service operations.
- LDAP directories use a tree structure as the data model. Different types of databases, such as Microsoft Active Directory, use different tree structures. LDAP administrators typically customize the directory structure based on the requirements of their organization. Consult with your LDAP administrator for information about the directory that you are querying.

22. Output Service

The Output service lets you create documents in different formats, including PDF, laser printer formats, and label printer formats. Laser printer formats are PostScript and Printer Control Language (PCL). The following list specifies label printer formats:

- Zebra (ZPL)
- Intermec (IPL)
- Datamax (DPL)
- TecToshiba (TPCL)

A document can be sent to a network printer, a local printer, or a file located on the file system. The Output service merges XML form data with a form design to generate a document. The Output service can generate a document without merging XML form data into the document. However, the primary workflow is merging data into the document. (See *Formdata*.)

NOTE: A form design is typically created by using Designer. For information about creating form designs for the Output service, see Designer Help.

When using the Output service to merge XML data with a form design, the result is a non-interactive PDF document. A non-interactive PDF document does not let users enter data into its fields. In contrast, you can use the Forms service to create an interactive PDF form that lets users enter data into its fields. (See *FormsService*).

The following four Output service operations are available for use within Workbench:

- **generatePDFOuput2**: Merges a form design with data to generate a PDF document
- **generatePrintedOutput2**: Merges a form design with form data to generate a document to send to either a laser or a label network printer
- **sendToPrinter**: Prints a document at the specified printer
- **transformPDF**: Converts an interactive PDF form to a non-interactive PDF document

  NOTE: If you provide a flat PDF document as the input for the **transformPDF** operation, an error is returned indicating that the document is already flat. However, if the input document contains both interactive and flat elements, the transformPDF service converts the pages containing the interactive content, but does not return any error indicating the presence of flat elements.

NOTE: By default, deprecated methods are not displayed in Workbench. Enable the respective option to see deprecated methods.

In addition to using Workbench, you can programmatically interact with the Output service by using the Output Service API. (See Programming with AEM Forms.)

22.1. Using the Output service

For information about developing processes that use this service, see Workbench Help.

For information about initially setting up this service, see administration console Help.

Creating PDF documents

You can use the Output service to create PDF document that is based on a form design and XML form data. The PDF document is a non-interactive PDF document. That is, users cannot enter or modify form data. A basic Output workflow is to merge XML form data with a form design to create a PDF document. The following illustration shows the Output service merging a form design and XML form data to produce a PDF document.

Creating PDF/A documents

Although PDF/A is the standard for archiving PDF documents, you can store a PDF document if it meets your business requirements. However, a PDF document can contain content that becomes outdated. For example, a URL that references a website can become outdated in the future. The purpose of the PDF/A standard is to establish a PDF file that has the potential for long-term endurance.

The following illustration shows the Output service merging a form design and XML form data to produce a PDF/A document.

NOTE: The AIIM website has a PDF/A FAQ section at http://www.aiim.org/documents/standards/19005-1_FAQ.pdf.

Sending documents to printers

You can use the Output service to send laser print formats and label print formats to a printer. For a list of supported formats, see *OutputService*.

NOTE: Label printer manufactures have devices that use different print languages. Not all Zebra printers use ZPL; therefore, ensure that your device can support the specific print language.

The following illustration shows the Output service sending documents to network printers.

The Output service supports the following printing access mechanisms:

Direct accessible printer:

A printer that is installed on the same computer is called a *direct accessible printer*, and the computer is named *printer host*. This type of printer can be a local printer that is connected to the computer directly.

Indirect accessible printer:

Technologies such as the common UNIX® printing system (CUPS) and the Line Printer Daemon (LPD) protocol are available. The printer that is installed on a print server is accessed from other computers. To access an indirect accessible printer, the print server's IP or host name must be specified. Using this mechanism, you can send a document to an LPD URI when the network has an LPD running. This mechanism lets you route the document to any printer that is connected to the network that has an LPD running.

When using this method to send a document to a printer, specify one of these printing protocols:

CUPS:

A printing protocol named *common UNIX printing system.*This protocol is used for UNIX operating systems and enables a computer to function as a print server. The print server accepts print requests from client applications, processes them, and sends them to configured printers. On the IBM AIX® operating system, usage of CUPS is not recommended.

DirectIP:

A standard protocol for remote printing and managing print jobs. This protocol can be used locally or remotely. Print queues are not required.

LPD:

A printing protocol named *Line Printer Daemon protocol* or *Line Printer Remote (LPR)* protocol. This protocol provides network print server functionality for UNIX-based systems.

Shared Printer:

A printing protocol that enables a computer to use a printer that is configured for that computer. This protocol does not work if the Generate PDF service is installed and AEM Forms on JEE is installed on the Windows Server operating system. This issue is applicable only for the Windows Server 2008 operating system. In this situation, use a different printer protocol.

CIFS:

The Output service supports the Common Internet File System (CIFS) printing protocol. (See *ImprovingtheperformanceoftheOutputservice*.)

The following table lists various input values, printing access mechanisms, and the results.

Access mechanism	Print server URI	Printer name	Result
SharedPrinter	Any	Null	Exception: Required argument `sPrinterName` cannot be null.
SharedPrinter	Any	Invalid	Output service throws an exception stating that the printer cannot be found.
SharedPrinter	Any	Valid	Successful print job.
LPD	Null	Any	Output service throws an exception stating that the required argument `sPrintServerUri` cannot be null.
LPD	Invalid	Null	Output service throws an exception stating that the required argument `sPrinterName` cannot be null.
LPD	Invalid	Not null	Output service throws an exception stating that `sPrintServerUri` is not found.
LPD	Valid	Invalid	Output service throws an exception stating that the printer cannot be found.
LPD	Valid	Valid	A successful print job.
CUPS	Null	Any	Output service throws an exception stating that the required argument `sPrintServerUri` cannot be null.
CUPS	Invalid	Any	Output service throws an exception stating that the printer cannot be found.
CUPS	Valid	Any	Successful print job.
DirectIP	Null	Any	Output service throws an exception stating that the required argument `sPrintServerUri` cannot be null.
DirectIP	Invalid	Any	Output service throws an exception stating that the printer cannot be found.
DirectIP	Valid	Any	Successful print job.
CIFS	Valid	Null	Successful print job.

Access mechanism	Print server URI	Printer name	Result
CIFS	Invalid	Any	Output service throws an unknown error while printing using CIFS.
CIFS	Null	Any	Output service throws an exception stating that the required argument `sPrintServerUri` cannot be null.

NOTE: To print on remote network printers on Windows, it is easier to use the CIFS protocol than a shared printer. The CIFS protocol can be used with remote Windows print servers. (See *Improvingtheperformanceoftheoutputservice*.)

Running a service on Windows

If you install AEM Forms on JEE using the turnkey installation for JBoss on Windows, JBoss runs in the context of the Local System account. Services that run in this context do not have access to network resources such as printers because the services are not authenticated on the network. If you use the Output Installation Verification Sample (Output IVS) to send a document to a network printer, the following error message is displayed:

```
Printer \\server\queue not found
```

To solve this problem, enable JBoss to run in the context of a valid user. To perform this task, change the properties of the JBoss service by clicking the Log On tab and selecting This Account. Specify a valid user name and password.

NOTE: This issue is applicable only when using the SharedPrinter access mechanism.

Sending the document to a network printer

You can send a document to a line printer daemon (LPD) URI when the network has an LP daemon running. You can route the document to any printer that is connected to the network. This printer can exist on a separate computer.

After you reference XML form data and set print run-time options, you can invoke the Output service. The Output service sends the document to a network printer. It is recommended that you use the CIFS protocol when possible. A shared printer can be used when a printer is locally installed.

A PostScript file created from a form design that contains a custom page size does not print correctly. To correct this issue, configure the printer to handle custom sizes. Each printer has its own way of handling custom sizes. Some printers allow you to configure the page size, media type, input trays, and so on. Read your printer's documentation to learn how to configure your printer to handle custom sizes.

When you send a document to a printer, consider the following factors:

- Set `PrinterProtocol` to `SharedPrinter`, set `ServerURI` to a blank value, and set `PrinterName` to a value that specifies the printer path (for example, `\\server12r-nt\HP LaserJet 8150 PCL 6 Tower II Level 5`).
- You can retrieve the value for `PrinterName` by using the path where the printer is installed. For example, assume that the printer is on the `server12r-nt` server. To obtain the name of the

printer, select Start, Printers and Faxes, right-click the specific printer, and select Properties. On the General Tab, the text box displays the printer name.

- One reason that an issue can occur when printing with a SharedPrinter is that the login identifier is not correct. (See RunningaserviceonWindows.)

Processing batch data to create multiple documents

The Output service can create separate documents for each record within an XML batch data source. The Output service can also create a single document that contains all records (this functionality is the default). Assume that an XML data source contains ten records and you instruct the Output service to create a separate document for each record (for example, PDF documents). As a result, the Output service generates ten PDF documents.

The following illustration also shows the Output service processing an XML data file that contains multiple records. However, assume that you instruct the Output service to create a single PDF document that contains all data records. In this situation, the Output service generates one document that contains all of the records.

The following illustration shows the Output service processing an XML data file that contains multiple records. Assume that you instruct the Output service to create a separate PDF document for each data record. In this situation, the Output service generates a separate PDF document for each data record.

The following XML data shows an example of a data file that contains three data records.

```
<?xml version="1.0" encoding="UTF-8"?>
<batch>
<LoanRecord>
<mortgageAmount>500000</mortgageAmount>
<lastName>Blue</lastName>
<firstName>Tony</firstName>
<SSN>555666777</SSN>
<PositionTitle>Product Manager</PositionTitle>
<Address>555 No Where Dr</Address>
<City>New York</City>
<StateProv>New York</StateProv>
<ZipCode>51256</ZipCode>
<Email>TBlue@NoMailServer.com</Email>
<PhoneNum>555-7418</PhoneNum>
```

```
<FaxNum>555-9981</FaxNum>
<Description>Buy a home</Description>
</LoanRecord>
<LoanRecord>
<mortgageAmount>300000</mortgageAmount>
<lastName>White</lastName>
<firstName>Sam</firstName>
<SSN>555666222</SSN>
<PositionTitle>Program Manager</PositionTitle>
<Address>557 No Where Dr</Address>
<City>New York</City>
<StateProv>New York</StateProv>
<ZipCode>51256</ZipCode>
<Email>SWhite@NoMailServer.com</Email>
<PhoneNum>555-7445</PhoneNum>
<FaxNum>555-9986</FaxNum>
<Description>Buy a home</Description>
</LoanRecord>
<LoanRecord>
<mortgageAmount>700000</mortgageAmount>
<lastName>Green</lastName>
<firstName>Steve</firstName>
<SSN>55566688</SSN>
<PositionTitle>Project Manager</PositionTitle>
<Address>445 No Where Dr</Address>
<City>New York</City>
<StateProv>New York</StateProv>
<ZipCode>51256</ZipCode>
<Email>SGreeb@NoMailServer.com</Email>
<PhoneNum>555-2211</PhoneNum>
<FaxNum>555-2221</FaxNum>
<Description>Buy a home</Description>
</LoanRecord>
</batch>
```

NOTE: Notice that the XML element that starts and ends each data record is `LoanRecord`.

Set PDF run-time options

Set the following run-time options for the Output service to successfully process batch data and create multiple files based on an XML data source:

Many Files:

Specifies whether the Output service creates a single document or multiple documents. You can specify `true` or `false`. To create a separate document for each data record in the XML data source, specify `true`.

File URI:

Specifies the location of the files that the Output service generates. For example, assume that you specify C:\\Adobe\forms\Loan.pdf. In this situation, the Output service creates a file named *Loan.pdf* and places the file in the C:\\Adobe\forms folder. When multiple files exist, the file names are Loan0001.pdf, Loan0002.pdf, Loan0003.pdf, and so on. The documents are placed on the server hosting AEM Forms on JEE, not the client computer.

Record Name:

Specifies the XML element name in the data source that separates the data records. For example, in the example XML data source that is shown, the XML element that separates data records is called `LoanRecord`. (Instead of setting the Record Name run-time option, you can set Record Level by assigning it a numeric value that indicates the element level containing data records. However, you can set only the Record Name or the Record Level; you cannot set both values. In the above XML, the record level to create multiple documents is 2.

Incremental loading

When the Output service processes batch records, it reads data that contains multiple records in an incremental manner. That is, the Output service reads the data into memory and releases the data as the batch of records is processed. The Output service loads data in an incremental manner when either one of two run-time options is set. If you set the Record Name run-time option, the Output service reads data in an incremental manner. Likewise, if you set the Record Level run-time option to 2 or greater, the Output service reads data in an incremental manner.

You can control whether the Output service performs incremental loading by using the `PDFOutputOptionsSpec` or the `PrintedOutputOptionSpec` object's `setLazyLoading` option. You can specify the value `false`, which turns off incremental loading.

If the Output service cannot perform incremental data loading, the Output service writes the following warning message in the log file of the J2EE application server hosting AEM Forms on JEE:

```
*  2007-11-01 11:51:23,215 WARN  [com.adobe.document.XMLFormService]
$$$/com/adobe/document/xmlform/msg.XFA=Unable to perform an incremental
data load. Performing a full data load.
```

NOTE: If the Output service does not read data in an incremental manner, the entire batch data file is read into memory. This behavior can have a detrimental effect on the performance of the Output service. (See *ImprovingtheperformanceoftheOutputservice*.)

Creating search rules

You can create pattern matching rules that result in the Output service examining input data and using different form designs based on the data content. For example, if the text *mortgage* is located within the input data, the Output service can use a form design named Mortgage.xdp. Likewise, if the text *automobile* is located in the input data, the Output service can use a form design that is saved as Automobile-Loan.xdp. The following illustration shows the Output service generating a PDF file by processing an XML data file and using one of many form designs.

The Output service can generate document packages, where multiple records are provided in the data set and each record is matched to a form design. The Output service generates a single document made up of comprised of multiple form designs.

To define pattern matching rules, define one or more text pattern that the Output services searches for in the input data. For each text pattern that you define, specify a corresponding form design that is used if the text pattern is located within the data. If a text pattern is located, the Output service uses the corresponding form design to generate the output. An example of a text pattern is *mortgage*.

Flattening interactive PDF documents

You can use the Output service to transform an interactive PDF document (for example, a form) to a non-interactive PDF document. An interactive PDF document lets users enter or modify data located in the PDF document fields. The process of transforming an interactive PDF document to a non-interactive PDF document is called *flattening*. When a PDF document is flattened, a user cannot modify the data located in the document's fields. One reason to flatten a PDF document is to ensure that data cannot be modified.

You can flatten the following types of PDF documents:

- Interactive PDF documents created in Designer (that contain XFA streams).
- Acrobat PDF forms

If you attempt to flatten a non-interactive PDF document, an exception occurs.

Retaining Form State

An interactive PDF document contains various elements that constitute a form. These elements may include fields (to accept or display data), buttons (to trigger events), and scripts (commands to perform a specific action). Clicking a button may trigger an event that changes the state of a field. For example, choosing a gender option may change the color of a field or the appearance of the form. This is an example of a manual event causing the form state to change.

When such an interactive PDF document is flattened using the Output service, the state of the form is not retained. To ensure that the state of the form is retained even after the form is flattened, set the Boolean value `retainPDFFormState` to `True` to save and retain the state of the form.

Assembling custom form designs for Output

You can use the Output and Assembler services together to create a highly customized document. The Assembler service assembles an XDP document (a form design) that is made up of multiple XDP forms and inserted fragments located in multiple XDP files. The assembled XDP document is passed to the Output service, which creates a PDF document.

You can specify the containers to be included when a PDF is generated from the XDP documents. In addition, when you use the Assembler service, you can resolve the images referenced in the source XDP documents. You can specify whether to resolve all, none, all relative or all absolute references. By default, none of the references are resolved.

The following illustration shows this workflow.

For information about creating a form design from fragments, see *Guidelines for Dynamically Assembling Customized Forms and Documents*.

Using Output IVS to check your Printer Description file (PDL)

Output IVS is a sample application for testing the Output service. Using this sample application, you can generate documents and test form designs and data sets. You can also print documents by using laser and label printers.

Administrators can use Configuration Manager to deploy Output IVS. They can also manually deploy it. (See the installation document specific to your application server, from the AEM Forms Documentation page)

To open the Output IVS application, navigate to http://*[server_name:port_number]*/OutputIVS.

To change settings that Output IVS uses, click the Preferences link on the AEM Forms on JEE Output banner. Here are some of the settings you can specify from the Preferences window:

- Locations that Output IVS obtains form, data, XDC, and companion files from. The locations can be URLs, a repository, or an absolute reference from a folder on the computer that hosts AEM Forms on JEE. Repository locations can be specified as either *repository:/* or *repository:///*.
- Common format and options, such as whether Output IVS creates a single output stream or includes metadata.
- Print options, such as duplex and number of copies.
- Administrator credentials.

To specify characteristics about your job and to submit your job, click the Test Output link on the AEM Forms on JEE Output banner. Here are some of the settings you can specify from the Test Output window:

- Output format, such as PDF, PDFA-1/a and ZPL 300 DPI

- File selection specifies the form, data, XDC, and companion files to use in the test. Use the companion file for these settings:
  - Pattern-matching rules for mapping data elements to different templates
  - Batch settings for initializing lazy loading
  - Many of the other settings you can specify by using the Preferences window
- Output location information such as server file or printer name
- Issue request sends the request to the Output service

To view or delete files used by Output IVS, click the Maintenance link on the AEM Forms on JEE Output banner. You can delete only the files that you added to Output IVS. You cannot delete files that are installed with Output IVS. The Maintenance window lists the form, data, XDC, and companion files in the locations that are specified using the Preferences window. You can also use the Browse buttons to upload files from your own computer

To see the complete Help for Output IVS, click the Help link on the AEM Forms on JEE Output banner.

22.2. Considerations for the Output service

Retaining Legacy Appearance of PDF Forms in AEM Forms on JEE

In AEM Forms on JEE, PDF forms rendered with the assistance of Forms, Output, or Form Data Integration (FDI) services have the appearance that is close to PDF forms in Acrobat XI. However, if you are upgrading from the previous version of AEM Forms on JEE, and would like to retain the older appearance of PDF forms, you can set a service-level configuration option for the Forms, Output, and FDI services. For the Forms, Output, or FDI service, when the value of this option (Appearance Compatibility Mode) is set to **9**, all PDF forms rendered using the service appear in the legacy appearance. Any other value set for this option results in generated PDF forms appearing with the look and feel enabled by AEM Forms on JEE.

This option can be set from the administration console (see the ManagingServices topic in the *administration help*) or from Workbench (see Workbench Help).

Form data

The Output service accepts both a form design that is typically created in Designer and XML form data as input. To populate a document with data, an XML element must exist in the XML form data for every form field that you want to populate. The XML element name must match the field name. An XML element is ignored if it does not correspond to a form field or if the XML element name does not match the field name. It is not necessary to match the order in which the XML elements are displayed. The important factor is that the XML elements are specified with corresponding values.

Consider the following example loan application form.

To merge data into this form design, create an XML data source that corresponds to the form. The following XML represents an XML data source that corresponds to the example mortgage application form.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <xfa:datasets xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/">
- <xfa:data>
- <data>
- <Layer>
<closeDate>1/26/2007</closeDate>
<lastName>Johnson</lastName>
<firstName>Jerry</firstName>
<mailingAddress>JJohnson@NoMailServer.com</mailingAddress>
<city>New York</city>
<zipCode>00501</zipCode>
<state>NY</state>
<dateBirth>26/08/1973</dateBirth>
<middleInitials>D</middleInitials>
<socialSecurityNumber>(555) 555-5555</socialSecurityNumber>
<phoneNumber>5555550000</phoneNumber>
</Layer>
- <Mortgage>
<mortgageAmount>295000.00</mortgageAmount>
<monthlyMortgagePayment>1724.54</monthlyMortgagePayment>
<purchasePrice>300000</purchasePrice>
<downPayment>5000</downPayment>
<term>25</term>
<interestRate>5.00</interestRate>
</Mortgage>
</data>
</xfa:data>
</xfa:datasets>
```

Supported document types

For complete access to the rendering capabilities of the Output service, it is recommended that you use an XDP file as input. In some cases, a PDF file can be used. However, using a PDF file as input has the following limitations:

- A PDF document that does not contain an XFA stream cannot be rendered as PostScript, PCL, or ZPL. The Output service can render PDF documents with XFA streams (that is, forms created in Designer) into laser and label formats. If the PDF document is signed, certified, or contains usage rights (applied using the Acrobat Reader DC extensions service), it cannot be rendered to these print formats.
- Run-time options such as PDF version and tagged PDF are not supported for Acrobat forms. They are valid for PDF forms that contain XFA streams; however, these forms cannot be signed or certified.

Signature fields

If a form design contains a signature field and you set the `renderAtClient` run-time option to `true`, an unsigned signature field is not retained in the resultant PDF document. However, if this run-time option is set to `false`, an unsigned signature field is retained in the resultant PDF document.

Flattening a digitally signed PDF document

If you try to flatten an interactive XFA PDF document that contains a signed signature field and set the `renderAtClient` run-time option to `true`, the resultant PDF document is not flattened. That is, the document remains interactive and the fields within the PDF document let users enter and modify data. This behavior occurs because the `renderAtClient` run-time option should be used when the input to an Output service operation is a form design (an XDP file), not a PDF file. To successfully flatten a digitally signed PDF document, set the `renderAtClient` run-time option to `false`.

Email support

For email functionality, you can create a process in Workbench that uses the Email service. A process represents a business process that you are automating. (See Workbench Help.)

Maximizing throughput

Output can process large XML data files that contain many records. You can instruct the Output service to create an output file for each record or a single output file that contains all records. (See *Processing-batchdatatocreatemultipledocuments*.)

Throughput limitations are not discreet and vary depending of the complexity of the form design, available memory, options chosen, and other activity. Errors that are generated when such a limitation is reached can identify lack of memory as being the problem. Be aware of this limitation, and use the Output service within number guidelines for safe page generation.

For a medium complexity form on a typical entry-level business server, the following approximate number guidelines for safe page generation were observed.

Format	Batch records/many small documents	Big record/one big document
PDF (not tagged) and PDF/a-1b	10000 pages	500 pages
	2500 pages	250 pages
PostScript	10000 pages	500 pages
PCL (using TrueType or printer-resident fonts)	10000 pages	500 pages

Select tagging for PDF only if it is required; tagged files are substantially larger than non-tagged files and take longer to render. For PCL printing, the use of Microsoft OpenType® fonts in form designs formats properly but decreases throughput, generates a larger output file, and does not deliver optimal performance. For increased performance, use or map properly licensed TrueType fonts or use printer-resident fonts.

When processing large data files or operating on a busy server, increase the Output service time-out; the default time-out is 180 seconds. To change the time-out value, ensure that hardware servers have adequate memory and the memory is available to the Java application server heap. For information about changing the time-out value, see administration console Help.

When processing a large record or one large document case, throughput is maximized when the XML data is structured so that a `recordLevel` run-time option of 2 can be used. For example, instead of structuring your data file in this manner:

```
<datafile>
<field>123</field>…
</datafile>
```

Structure it this way, adding another level:

```
<datafile>
<record>
<field>123</field>
…
</record>
</datafile>
```

Printable areas

The default 0.25-inch nonprintable margin is not exact for label printers and varies from printer to printer and from label size to label size. It is recommended that you keep the 0.25-inch margin or reduce it. However, it is recommended that you do not *increase* the nonprintable margin. Otherwise, information in the printable area does not print properly.

Always ensure that you use the correct XDC file for the printer. For example, avoid choosing an XDC file for a 300-dpi printer and sending the document to a 200-dpi printer.

Scripts

A form design that is used with the Output service can contain scripts that run on the server. Ensure that a form design does not contain scripts that run on the client. For information about creating form design scripts, see Designer Help.

Working with fonts

The following topics discuss how the Output service handles fonts within a generated document:

- Ensuring that fonts are available to AEM Forms on JEE
- Using printer-resident fonts
- Font mapping

Ensuring that fonts are available to AEM Forms on JEE

Ensure that a font used within a form is available for use on the J2EE application server hosting AEM Forms on JEE. For example, consider the following scenario. A form designer adds a font to the font directory that Designer uses and creates a form that uses that font on a separate computer.

In order for the Output service to use the font, place it in the Customer fonts directory. Specify the location of the Customer fonts directory by using administration console. If the Customer fonts directory does not exist, create a directory on the J2EE application server hosting AEM Forms on JEE. Place the font in the new Customer fonts directory.

Specify the location of the Customer fonts directory

1) In administration console, click Settings > Core System Settings.
2) In the Core System view, click Configurations.
3) Specify the location of the Customer fonts directory in the Location of the Customer Fonts directory text input box.
4) Click OK.
5) Restart the J2EE application server on which AEM Forms on JEE is installed.

Using printer-resident fonts

The Output service can generate documents that contain resident fonts. XDC files that are available with the Output service have a common list of printer-resident fonts defined. You do not need to edit an XDC file to use these fonts. However, you can modify an XDC file by using AEM Forms on JEE XDC Editor. (See XDC Editor Help.)

printer-resident fonts are stored inside a printer, either in a special hardware cartridge or built in the printer's memory. For PostScript printers, only the name of the font is required. For PCL printers, it is

necessary to define an escape sequence of characters. printer-resident font names can be obtained from the printer's front panel or by using the printer's built in web server (if available).

To use printer-resident fonts, you can perform these tasks:

1) Identify the name of the printer-resident font to use.
2) Specify the printer-resident font name in the form design.
3) Locate the most suitable XDC file to use for your printer based on the type of output to render (PostScript or PCL) and make the appropriate changes to the XDC file.
4) Test the output.

NOTE: For information about performing these tasks, see XDC Editor Help.

Font mapping

If a font is installed on a client computer, it is available in the drop-down list in Designer. If the font is not installed, it is necessary to specify the font name manually. The "Permanently replace unavailable fonts" option in Designer can be off. Otherwise, when the XDP file is saved in Designer, the substitution font name is written to the XDP file. This means that the printer-resident font is not used.

To design a form that uses printer-resident fonts, choose a typeface name in Designer that matches the fonts that are available on the printer. A list of fonts that are supported for PCL or PostScript are located in the corresponding device profiles (XDC files). Alternatively, font mapping can be created to map nonprinter-resident fonts to printer-resident fonts of a different typeface name. For example, in a PostScript scenario, references to the Arial font can be mapped to the printer-resident Helvetica typeface.

Two types of OpenType fonts exist. One type is a TrueType OpenType font that PCL supports. The other is CFF OpenType. PDF and PostScript output supports embedded Type-1, TrueType, and OpenType fonts. PCL output supports embedded TrueType fonts.

Type-1 and OpenType fonts are not embedded in PCL output. Content that is formatted with Type-1 and OpenType fonts is rasterized and generated as a bitmap image that can be large and slower to generate.

Downloaded or embedded fonts are automatically substituted when generating PostScript, PCL, or PDF output. This means that only the subset of the font glyphs that are required to properly render the generated document is included in the generated output.

For information about creating form designs for the Output service, including handling fonts, see *Designing Forms for AEM Forms on JEE Output*.

CIFS printing support

The Output service supports Common Internet File System (CIFS) printing. A printer administrator must make a printer accessible to the CIFS protocol. On computers running Windows, this step is done by sharing the printer or the net share command. On UNIX-like systems, this step is done by using the Samba server.

If the printer is secured, the printer administrator may provide the AEM Forms administrator with the user name and password that are required to print to the printer queue. To access the printer by using the user name and password, the AEM Forms administrator must register them by using AEM Forms on

JEE trust store. The Output service looks up the user name and password for a given UNC path by using it and its parent components as the profile name.

Assume that the AEM Forms administrator registers `\\print-server.domain.name.com\printer` as a profile in the AEM Forms on JEE trust store. In this situation, the Output service searches for the credentials of a printer by looking up aliases in the Trust Store in the following order:

```
\\print-server.domain.name.com\printer
\\print-server.domain.name.com
\\domain.name.com
\\name.com
\\com
\\
```

The Output service sends the print job to the first matching entry.

NOTE: AEM Forms on JEE supports NTLM v2 in CIFS. Kerberos5 is not supported.

Assume that a AEM Forms user invokes the `sendToPrinter` operation, passes the printer path as the server name parameter, and leaves the printer name parameter blank. The required user name and password values are resolved by using AEM Forms on JEE Trust Store profiles.

When using the CIFS protocol, the printer queue is shared as a resource by a server. To print a file, the following tasks are performed:

- The client authenticates itself to the server for the specific resource (for example, the shared printer queue on server).
- After authentication, the print job is sent to the printer.

Working with device profile files (XDC file)

A device profile (XDC file) is a printer description file in XML format. This file enables the Output service to output documents as laser or label printer formats. The Output service uses the XDC files including these:

- hppcl5c.xdc
- hppcl5e.xdc
- ps_plain_level3.xdc
- ps_plain_mt.xdc
- ps_plain.xdc
- zpl203.xdc
- zpl300.xdc
- zpl600.xdc

It is not necessary to modify these files for the Output service to create documents. However, you can modify them to meet your business requirements. (See XDC Editor Help.)

NOTE: See XDC Editor Help for a complete listing of XDC files.

In addition, the following files are sample XDC files that support the features of specific printers, such as resident fonts, paper trays, and stapler. The purpose of these samples is to help you understand how to

set up your own printers by using device profiles. The samples are also a starting point for similar printers in the same product line.

Printer tray selection

When sending a document to a printer by using the Output service, you can specify the printer tray that is used. Designer distinguishes between paper (page) size and which input tray on the printer provides the requested paper. This functionality accommodates scenarios where a printer has different types of a particular paper size loaded into different input trays. This functionality allows a document to select paper from individual trays on a per-page basis.

Designer does not expose paper size and input tray selection as two distinct properties. Instead, a Designer master page can be associated with a paper type selected from a set of supported paper types defined in the Designer.xdc device profile. Within the device profile, each paper type can be configured to select paper from a particular input tray. (See *Workingwithdeviceprofilefiles(XDCfile)*.)

Designer provides duplicates of common paper types for Letter and Legal paper sizes, specifically to accommodate paper tray selection. The XDC Editor within Workbench is used to map or assign a physical input tray to a paper type. The names provided in Designer covers most tray-selection needs. The deployed device profile can be modified so that the Letter Color paper type causes the printer to select paper loaded into a secondary input tray.

The Output service matches paper types that are used in the form, by name, against paper types that are defined in the device profile deployed to the server. Only the deployed device profile is modified to ensure the appropriate input tray selection.

Although the provided paper types are adequate, it is possible to create additional paper types. Additional paper types can be used in the XDC Editor by creating a paper type name. However, it must exist in Designer to be used. To add paper types, the Designer.xdc file must be hand-edited, copying an existing entry of the correct paper size and changing the name as required.

The concept of printer trays is not applicable to PDF. That is, you cannot specify a particular printer tray when printing a PDF document. The selection at the printer is based on page size, and the first non-secured tray that matches the required page size is used. If no match to the size is found, a manual feed is requested.

For PCL documents, each master page in the XDP file (the form design created in Designer) is mapped to a paper type. The paper type is, in turn, mapped to an entry in the XDC file. It is important that you use caution because Designer may turn the literal that you view in the paper type list into a slightly different literal. Check the XML source to get the precise literal that is used in the XDC file.

In the XDC file, the paper type appears in the stock column. You can modify the entry to show the tray number that contains the paper type you want to use. However, the tray number that you see on the printer itself (for example, tray 1, tray 2, and so on) may not be the correct device number that the printer understands. Review the printer reference manual to ensure that you have the tray number correctly stated.

If the stated tray number is not valid or carries a page size that does not match the requested page size, the printer reverts to the first non-secured tray that represents the correct paper size.

For PostScript documents, the printer tray selection process is the same as for PCL documents. To select the tray by media type, keep the Input Tray Number column in the XDC file blank and enter the media type in the Input Tray Type column. It is assumed that the printer is configured to recognize the media type.

NOTE: A PostScript file that is created from a form design that contains a custom page may not print. In this situation, configure the printer to handle custom sizes. Each printer has its own way of handling custom sizes. Some printers allow you to configure the page size, media type, input trays, and so on. See your printer's documentation to learn how to configure your printer to handle custom sizes.

Paper handling

Designer exposes control over duplex printing in two ways depending on whether the form is used to generate a PDF document and printed from Adobe Reader or Acrobat. Or whether the form is printed directly to a PCL or PostScript device.

When the form is intended to generate a PDF document, settings that relate to how the PDF document is printed can be configured from the Form Properties dialog box within Designer. These settings include the number of copies to print and the duplexing setting. Subsequent printing of a PDF document from Adobe Reader or Acrobat uses these settings.

Duplexing can also be specified by using the pagination property of the Output service operations that are available within Workbench. Duplexing can also be specified by using the Output service Java and web service API. Additional capabilities are available in Designer to design forms that adjust according to simplex and duplex printing scenarios.

Master pages can be assigned to odd-numbered (front side) or even-numbered (back side) printed pages. Different master pages can be designed for the front and back pages. The Output service automatically selects the appropriate master page, depending on whether it is currently printing on the front or back of the page. A common use case is to create master pages that place the page count on the left side or right side of the page. Assign the master pages as odd or even to ensure that the page count is always on the inside or outside of a duplex-printed document.

Working with the XCI configuration file

The Output service uses an XCI configuration file to perform tasks, such as embedding a font into a document. Although this file contains settings that can be set, it is not typical to modify this value. The default.xci file is located in the svcdata\XMLFormService folder.

For example, assuming that:

- AEM Forms on JEE is installed on JBoss applications server, the full path is as follows:
  `[AEM Forms Install location]\jboss\standalone\svcdata\XMLFormService`
- AEM Forms on JEE is installed on WebLogic application server, the full path is as follows:
  `[appserver root]\user_projects\domains\[client_domain_name]\adobe\server 1\XMLFormService`
- AEM Forms on JEE is installed on WebSphere application server, the full path is as follows:

```
[appserver root]\AppServer\profiles\[client_profile_name\installedApps\a
dobe\server1\XMLFormService
```

You can pass a modified XCI file while performing an Output service operation. When doing so, create a copy of the default file, change only the values that requires modification to meet your business requirements, and use the modified XCI file.

The Output service starts with the default XCI file (or the modified file). Then it applies values that are specified in Workbench property sheets or specified by using the Output Service API. These values override XCI settings. For example, values specified in a Workbench property sheet overrides XCI values.

The following table specifies XCI options.

XCI option	Description
config/present/pdf/creator	Identifies the document creator using the Creator entry in the Document Information dictionary. For information about this dictionary, see the *PDF Reference* guide.
config/present/pdf/producer	Identifies the document producer using the Producer entry in the Document Information dictionary. For information about this dictionary, see the *PDF Reference* guide.
config/present/layout	Controls whether the output is a single panel or paginated.
config/present/pdf/compression/level	Specifies the degree of compression to use when generating a PDF document.
config/present/pdf/fontInfo/embed	Controls font embedding in the output document. Set the value of the embed tag to 1 to embed fonts in the output document.
config/present/pdf/scriptModel	Controls whether XFA-specific information is included in the output PDF document.
config/present/common/data/adjustData	Controls whether the XFA application adjusts the data after merging.
config/present/pdf/renderPolicy	Controls whether the generation of page content is done on the server or deferred to the client.
config/present/common/locale	Specifies the default locale used in the output document.
config/present/destination	When contained by a `present` element, specifies the output format. When contained by an `openAction` element, specifies the action to perform upon opening the document in an interactive client.
config/present/output/type	Specifies either the type of compression to apply to a file or the type of output to produce.
config/present/common/temp/uri	Specifies the Form URI.

XCI option	Description
config/present/common/template/base	Supplies a base location for URIs in the form design. When this element is absent or empty, the location of the form design is used as the base.
config/present/common/log/to	Controls the location that log data or output data is written to.
config/present/output/to	Controls the location that log data or output data is written to.
config/present/script/currentPage	Specifies the initial page when the document is opened.
config/present/script/exclude	Informs AEM Forms on JEE which events to ignore.
config/present/pdf/linearized	Controls whether the output PDF document is linearized.
config/present/script/runScripts	Controls which set of scripts AEM Forms on JEE executes.
config/present/pdf/tagged	Controls the inclusion of tags into the output PDF document. Tags, in the context of PDF, are additional information included in a document to expose the logical structure of the document. Tags assist accessibility aids and reformatting. For example a page number may be tagged as an artifact so that a screen reader does not enunciate it in the middle of the text. Although tags make a document more useful, they also increase the size of the document and the processing time to create it.
config/present/pdf/fontInfo/alwaysEmbed	Specifies a font that is embedded into the output document.
config/present/pdf/fontInfo/neverEmbed	Specifies a font that must never be embedded into the output document.
config/present/pdf/pdfa/part	Specifies the version number of the PDF/A specification that the document conforms to.
config/present/pdf/pdfa/amd	Specifies the amendment level of the PDF/A specification.
config/present/pdf/pdfa/conformance	Specifies the conformance level with the PDF/A specification.
config/present/pdf/version	Specifies the version of PDF document to generate

Embedding fonts

To demonstrate how to work with an XCI file, an XCI file is used to embed a font into a PDF document that the Output service generates. Embedding a font is relevant only for PDF output. For PCL and PostScript output, the font is automatically embedded if it is not resident on the printer. This setting reduces

the PDF file size by having the global embed setting off but providing a way to embed particular fonts into a document.

To embed a font into the resultant document, perform the following tasks:

1) Place the font into the windows\fonts folder that belongs to the computer that AEM Forms on JEE is deployed on.
2) Copy the XCI file located in the folder that is specified in this section, paste the XCI file in a new location, and rename it *custom.xci*. For example, use the location C:\XCI\custom.xci.
3) Edit the custom.xci file by creating an XML element named `fontInfo` (place it under the `pdf` element). Within the `fontInfo` element, add the following XML data:

   `<alwaysEmbed>[name of the new font]</alwaysembed>`
4) Modify the XCI URI option within Workbench or by using Output Service API.

Improving the performance of the Output service

The following list specifies performance factors to consider when working with the Output service:

- When invoking the Output service within a process, you can improve the performance by making the process a short-lived process. A long-lived process persists data in the AEM Forms on JEE database, resulting in a performance cost.
- You can turn off security for the Output service. By turning off security, the Output service does not authenticate a user when it performs an operation, therefore improving the performance of a given operation.
- Changing the caching option of the AEM Forms on JEE repository to manual optimizes performance. The default setting results in lower performance because the Output service assumes that a form (or fragments) have changed since the last time it was used.
- Forms that contain justified fields decrease performance; that is, justification degrades performance.
- Use of non-explicit data binding in form designs for batch processing results in decreased performance. (See *Processingbatchdatatocreatemultipledocuments*.)
- When working with batch processing and when possible, use incremental loading. (See the section on Incremental loading)
- When designing a form, use fonts that support a particular language to improve performance.

  NOTE: For information about processing large XML data files that contain many records, see *Maximizingthroughput*.

23. PDF Utilities Service

Use the PDF Utilities service to convert documents between PDF and XDP file formats, set and retrieve the save mode of PDF documents, and query information about a PDF document. For example, you can determine whether a PDF document contains comments or attachments.

23.1. Using the PDF Utilities service

You can accomplish the following tasks by using this service:

Clone PDF:

This feature is available only in Workbench. It replicates a PDF document. The resulting PDF document can be manipulated independently of the input PDF document. If a given PDF document is passed to multiple services without cloning, the result may be difficult to use effectively.

For example, assume that a PDF document is passed to two services sequentially. When the first service modifies and returns the PDF document as a document value, the next service to use the document value detects modifications that the first service made.

After using the Clone PDF operation, you are assured that the input document value and the result document value are identical but distinct. Also, any future modification of either value is not reflected in the other object.

Multiple Clone of PDF:

This feature is available only in Workbench. It clones a PDF document a specified number of times. The resulting PDF documents are used independently of the input PDF document.

Convert PDF documents to XDP documents:

Converts a PDF document to an XDP file. For a PDF document to be successfully converted to an XDP file, the PDF document must contain an XFA stream in the dictionary.

Convert XDP documents to PDF documents:

Converts an XDP file to a PDF file. To successfully convert an XDP file to a PDF file, the XDP file must contain an encoded PDF packet.

Retrieve PDF document properties:

Performs queries on the specified PDF document and returns the results as a `PDFPropertiesResult` value. You can perform the following queries:

- Is a PDF Document
- Is a PDF Package
- Get the PDF Version
- Check for Attachments

- Check for Comments
- Recommended Acrobat Version
- Form Type
- Check for AcroForm
- Has a Fillable Form
- Is an XFA Document
- Get the XFA Version

Get PDF Save Mode:

Returns the save mode of a PDF document. The save mode represents the mode the PDF document is saved in. Also, the save mode specifies whether the request is considered a requirement or only a suggestion. Save mode values are not influenced by the PDF document content. The following values are possible PDF save-mode values:

- `FAST_WEB_VIEW`, which is used while viewing the PDF document online
- `INCREMENTAL`, which performs the save operation in the least amount of time
- `FULL`, which saves with fewer optimizations

Set PDF Save Mode:

Sets the save mode of a PDF document. The save mode represents the mode the PDF document is saved in. Also, the save mode specifies whether the request is considered a requirement or only a suggestion. Save mode values are not influenced by the PDF document content.The following values are possible PDF save-mode values:

- `FAST_WEB_VIEW`, which is used while viewing the PDF document online
- `INCREMENTAL`, which performs the save operation in the least amount of time
- `FULL`, which saves with fewer optimizations

For information about developing processes that use this service, see Workbench Help. For information about developing client applications that programmatically interact with this service, see Programming with AEM Forms.

Redact PDF:

24. Acrobat Reader DC extensions Service

The Acrobat Reader DC extensions service enables your organization to easily share interactive PDF documents by extending the functionality of Adobe Reader with additional usage rights. The Acrobat Reader DC extensions service works with Adobe Reader 7.0 or later. The service adds usage rights to a PDF document. This action activates features that are not usually available when a PDF document is opened using Adobe Reader, such as adding comments to a document, filling forms, and saving the document. Third-party users do not require additional software or plug-ins to work with rights-enabled documents.

When PDF documents have the appropriate usage rights added, recipients can do the following activities from within Adobe Reader:

- Complete PDF documents and forms online or offline, allowing recipients to save copies locally for their records and still keep added information intact
- Save PDF documents to a local hard drive to retain the original document and any additional comments, data, or attachments
- Attach files and media clips to PDF documents
- Sign, certify, and authenticate PDF documents by applying digital signatures using industry-standard public key infrastructure (PKI) technologies
- Submit completed or annotated PDF documents electronically
- Use PDF documents and forms as an intuitive development front end to internal databases and web services
- Share PDF documents with others so that reviewers can add comments by using intuitive markup tools. These tools include electronic sticky notes, stamps, highlights, and text strikethrough. The same functions are available in Acrobat.
- Support barcoded forms decoding.

  These special user capabilities are automatically activated when a rights-enabled PDF document is opened within Adobe Reader. When the user is finished working with a rights-enabled document, those functions are again disabled in Adobe Reader. They remain disabled until the user receives another rights-enabled PDF document.

  The following topics provide information about the tasks that you can perform and how to achieve the best results:

- *UsingtheAcrobatReaderDCextensionsservice*
- *ConsiderationsfortheAcrobatReaderDCextensionsservice*

RELATED LINKS:

Using the Acrobat Reader DC extensions service

Considerations for the Acrobat Reader DC extensions service

24.1. Using the Acrobat Reader DC extensions service

Applying usage rights to PDF documents

You can use the Acrobat Reader DC extensions service to apply usage rights to PDF documents. Usage rights pertain to functionality that is available in Acrobat Pro but not in Adobe Reader. Such functionality includes the ability to add comments to a document, fill forms, and save the document. PDF documents that have usage rights added are called *rights-enabled documents*. A user who opens a rights-enabled PDF document in Adobe Reader can perform the operations that are enabled for that document.

Here is a list of the usage rights and the actions that users can perform in Adobe Reader when each usage right is enabled:

Basic form fill-in:

Users can fill in form fields and save files locally.

NOTE: A message prompting users to fill the form is displayed to users of Adobe Reader 7. This message is not displayed in Adobe Reader 8.

Import and export form data:

Users can import and export form data as FDF, XFDF, XML, and XDP files. If you select this feature, Basic Form Fill-In is also automatically selected. When you select this usage right, also enable the Database and Web Service Connectivity usage right or the Embedded File Attachments usage right (depending on the usage rights required).

Submit outside web browser:

Users can submit form data by email or offline. If you select this feature, Basic Form Fill-In is also automatically selected.

Database and web service connectivity:

Users can access the database or call the web service that is defined within the form. If you select this feature, Basic Form Fill-In is also automatically selected.

Add, delete, and change form fields:

Users can add, delete, or modify fields on the form. If you select this feature, Basic Form Fill-In is also automatically selected.

Create pages from templates:

Users can spawn pages from the form template for forms that are created in Acrobat. Users create pages from template pages within the same form. (Available in XFA documents but when selected, the usage right is not applied to the final saved document.) If you select this feature, Basic Form Fill-In is also automatically selected.

2D barcode decoding:

Users can use 2D barcodes with third-party scan decode solutions. If you select this feature, Basic Form Fill-In is also automatically selected.

Digital signatures:

Users can digitally sign and save the PDF document. If this option is not selected, users can still validate, view, and print the document that has a digital signature.

Commenting:

Embedded file attachments:

Users can add, remove, modify, or export file attachments and files in a PDF package (formerly known as a PDF portfolio).

Methods for applying usage rights

- Use the Acrobat Reader DC extensions web application. (See Acrobat Reader DC extensions Help.)
- Use processes that you develop by using Workbench. (See Workbench Help.)
- Use the Acrobat Reader DC extensions Service API. (See Programming with AEM Forms.)

Removing usage rights from PDF documents

You can use the Acrobat Reader DC extensions service to remove usage rights from a rights-enabled document. Removing usage rights from a rights-enabled PDF document is also necessary in order to perform other AEM Forms on JEE operations on it. For example, you must digitally sign (or certify) a PDF document before you set usage rights. Therefore, to perform operations on a rights-enabled document, do these tasks:

- remove usage rights from the PDF document
- perform the other operations such as digitally signing the document
- reapply usage rights to the document

Methods for removing usage rights

- Use processes that you develop by using Workbench. (See Workbench Help.)
- Use the Acrobat Reader DC extensions Service API. (See Programming with AEM Forms.)

Retrieving credential information

You can use the Acrobat Reader DC extensions service to retrieve information about the credential that was used to apply usage rights to a rights-enabled PDF document. By retrieving information about a credential, you can obtain information such as the date after which the certificate is no longer valid.

Methods for getting credential information rights

- Use the Acrobat Reader DC extensions Service API. (See Programming with AEM Forms.)
- Use processes that you develop by using Workbench. (See Workbench Help.)

24.2. Considerations for the Acrobat Reader DC extensions service

Consider the following factors when developing an application that uses the Acrobat Reader DC extensions service.

Configuring the credential

To apply usage rights to PDF documents, you must have a valid credential. A credential may have been configured during the installation of AEM Forms on JEE. For information about configuring a credential, see Configuring credentials for use with Acrobat Reader DC extensions in the *administration console Help* or consult your application server administrator.

Order of operations

Any combination of encrypting, certifying, and applying usage rights to the same document must occur in the following order. These services must be invoked within a short-lived process:

1) Apply encryption (Encryption service) or apply a policy (Document Security service) to a document before you digitally sign the document (Signature service). A digital signature records the state of the file at the time of signing. Encrypting the document or applying a policy after you apply a signature changes the bytes in the file, causing the signature to appear invalid.
2) Certify a PDF document (Signature service) before you set usage rights (Acrobat Reader DC extensions service). If you certify a document after you apply usage rights, it invalidates the usage rights signature, therefore removing the usage rights from the document.
3) Digitally sign a PDF document (Signature service) after you set usage rights. Signing a PDF document after applying usage rights does not invalidate the usage rights signature.

   Also, you cannot encrypt a PDF document and apply a policy to the same PDF document, and you cannot apply a policy to an encrypted PDF document.

Adding usage rights to interactive forms

Sometimes it is necessary to add usage rights to a PDF document while working with interactive forms that are created in Designer. For example, you created an interactive form by using Designer and you want to use a form script to reference a data connection SOAP endpoint where form data is retrieved. Then you want Adobe Reader to display the form.

Without usage rights, Adobe Reader fails to set the data connection SOAP endpoint, resulting in the predefined endpoint (set in Designer) being used. To set the data connection SOAP endpoint by using a form script, you must add the enableFormsOnline usage right to the interactive form.

Adding usage rights to forms that populate fields with data

When designing a form for use with Adobe Reader 7.0 where the form uses web services or database calls to populate fields with data, apply additional usage rights to your form.

When designing forms for use with Adobe Reader 8.0, apply the following usage rights:

- Basic form fill-in
- Database and web service connectivity
- Add, delete, and change form fields

  When designing forms for use with Adobe Reader 7.0, apply the following usage rights:
- Basic form fill-in
- Import and export form data
- Database and web service connectivity
- Add, delete, and change form fields

  Without the "Import and export form data" usage right, calls made to web services or a database fail in Adobe Reader 7.0.

Opening rights-enabled PDF documents

Users who attempt to open a rights-enabled PDF document in versions of Adobe Reader earlier than 7.0 may not be able to access certain features. These versions of Adobe Reader display a message indicating that users should upgrade to the latest version. For example, if a user applies usage rights to an Acrobat 5.0-compatible PDF document, the PDF document is no longer compatible with that version of Acrobat (or Adobe Reader). The PDF document is compatible only with Adobe Reader 7.0 or later.

25. Repository Service (Superseded)

The repository provides storage capabilities. When developers create applications, they can deploy the assets in the repository instead of on a file system. The repository can contain the following assets:

- XML forms
- PDF forms (including Acrobat forms)
- form fragments
- images
- processes
- profiles
- policies
- SWF files
- DDX files
- XML schemas
- WSDL files
- test data.

  The repository tracks the version of each asset in a AEM Forms on JEE application. At run time, services can retrieve assets from the repository as part of completing an automated business process.

25.1. Using the Repository service

You can use the Repository service in a process to retrieve resources from the repository. The Repository Service API provides a number of additional operations that you can use to store and retrieve information from the repository. For example, you can obtain a list of files or retrieve specific files stored in the repository when a file is needed as part of processing an application. You can also programmatically deploy application files by using the Repository Service API.

For information about developing a process that uses this service, see Workbench Help. For information about developing a client application that programmatically interacts with this service, see Programming with AEM Forms.

Direct use of the Repository is reserved for backwards compatibility, in cases where deployments are using Repository-based applications.

IMPORTANT: The Repository Service should not be used with AEM Forms. Also, the Repository Service should not be used with the current Application Model.

You can use the Applications and Services pages in Administration Console to configure default properties for this service. (See administration console Help.)

26. Document Security Service

Using the Document Security service, you can create policies and apply those policies to documents. A *policy* is a collection of information that includes confidentiality settings and a list of authorized users. You can specify who can open a document, limit how they can use it, and monitor the document after it is distributed. You can also dynamically control access to a document and revoke access to the document.

The Document Security service protects PDF files and other file types such as these:

- FLV and F4V
- Microsoft Word, Excel, and PowerPoint

Policies can be applied to PDF documents programmatically using the Document Security Service API, or as part of a process. Policies can also be applied to documents by using a client application. The procedures for how to apply policies to PDF documents are described in detail in *Acrobat Help*. Applying policies by using other applications, such as Microsoft Office, is documented in the *Document Security Extension Help* for the application. You can download the document security Extension for Microsoft Office from the Adobe website.

NOTE: Document Security requires authentication over SSL connections. (See administration console Help.)

Administrators use the Document Security web pages to do the following tasks (see *administrationconsole Help*):

- Configure various Document Security service settings
- Manage policy-protected documents
- Create and manage policy sets and the policies that they contain
- Monitor events that Document Security records
- Manage invited and local users

You can use Workbench to develop processes to do the following tasks (see Workbench Help):

- Create policies from templates or update existing policies
- Apply policies to documents, switch policies, or remove policy security
- Get license identifiers, revoke licenses, or unrevoke licenses
- Get all policy set names or all policy names within a policy set
- Get policy by policy identifier
- Inspect a protected document
- Unlock policy-protected PDF documents

You can use the Document Security Service API to develop client applications to do the following tasks (see Programmingwith AEM Forms):

- Create, modify, or delete policies
- Apply policies to or remove policies from PDF documents
- Revoke or reinstate access to PDF documents

- Create watermarks
- Search for events
- Open policy-protected documents in batch mode
- Retrieve information about policy-protected documents

26.1. About policies

A *policy* contains information about authorized users and the confidentiality settings to apply to documents. Users can be any user in your organization. Users can also be people external to your organization who registered with Document Security or for whom the administrator created an account. If the administrator enables the user invitation feature, you can add new invited users to policies. When you add a new invited user, the Document Security service sends a registration email inviting the user to register.

The confidentiality settings you specify in a policy determine how the recipients can use the document. For example, you can specify whether recipients can print or copy text, make changes, or add signatures and comments to protected documents. The same policy can also specify different confidentiality settings for different users.

You can create and save any number of policies, using security settings that are appropriate for different situations and users.

Using Document Security, you can dynamically change the permissions on a document. It gives the person who secures the document the permission to change the confidentiality settings to revoke access to the document or to switch the policy. After distributing the document, the person who secured it can monitor how the document is being used and who is using it.

Policies are described by using Portable Document Rights Language (PDRL).

Programmatically applying policies

On a mass production environment such as generating monthly invoices for a telecom company, creating and applying policies that are specific to each document can become a resource-intensive process. In such cases, you can use the Document Security Java API to create and apply policies that are specific to users, rather than to documents based on abstract policies. The license generated for a user is later used for all documents that are accessible to the user.

Using the APIs, you create an abstract policy that is a policy templates with all policy attributes such as document security settings and usage rights, except the list of principals. Administrators can create any number of policies from the abstract policy with different principals who should have access to the documents. Changes made to the abstract policy do not affect the actual policies that are generated from the abstract policies.

In the case of monthly invoice generation of a telecom company, you create an abstract policy, users, and then generate the licenses for each user that is later applied to the documents for each user.

You cannot create the abstract policy from the Document Security Web pages. You can, however, administer the policies that you create from the abstract policy from the Document Security web pages. Policies

that are created using this method are identical in behavior to those created from Document Security web pages.

See Programming with AEM Forms for more information.

26.2. About policy sets

*Policy sets*are used to group a set of policies that have a common business purpose. Policy sets are generally made available to a limited number of users by specifying which users or groups within a domain can use the policies from the policy set to protect documents.

Each policy set can have one or more associated policy set coordinators. The *policy set coordinator* is an administrator or a user who has additional permissions. The policy set coordinator is typically a specialist in the organization, one who can best author the policies in a particular policy set. Depending on the permissions assigned to the policy set coordinator, they may also be able to perform the following actions:

- view events related to the policy set
- manage documents
- manage other policy set coordinators

Policy sets are created and deleted in the Document Security administration web pages by policy set administrators who have the appropriate permission.

When Document Security is installed, a default policy set is created called *Global Policy Set*. Policy set administrators can administer this policy set.

26.3. Security methods and technology

To ensure the confidentiality of documents that are protected by policies, Document Security implements three layers of security.

Authentication

All users are required to log in to interact with Document Security. Users must log in before performing the following tasks:

- Opening the Document Security web application in a web browser
- Securing documents with policies in a supported client application
- Opening policy-protected documents

When creating a policy, you can allow anonymous users to open policy-protected documents if that setting is enabled in the Document Security configuration settings. When you allow anonymous user access, users who do not have accounts can access the document, but they cannot log in to Document Security or use other policy-protected documents.

Methods of authentication

Document Security supports these methods of authentication:

Username/Password:

Users are prompted for their user name and password.

Kerberos (Acrobat on Microsoft® Windows® only):

Enables Acrobat or Adobe Reader users on a Windows platform to be transparently authenticated.

Smart card (Acrobat on Microsoft Windows only):

Enables Acrobat or Adobe Reader users on a Windows platform to be authenticated by using a smart card.

Internal users have corresponding user records in your organizational user directory, and those records are synchronized with the User Management database. Document Security authenticates internal users against the User Management database. Document Security also stores external user accounts in the database and uses the accounts to authenticate external users. For information about managing users, see Adding and configuring users in the *administration console Help*.

Extended authentication

Users are directed to an extended authentication provider URL, where additional authentication methods are deployed.

Certificate-based authentication

Users can use authentication modes enabled by certificates.

SAML authentication assertions

After users are initially authenticated and when Document Security receives subsequent messages from clients, Document Security uses SAML authentication assertions to verify the identity of the message sender. Security Assertion Markup Language (SAML) authentication assertions are used for authentication until the assertion expires or users terminate their session.

When users are initially authenticated by using their user name and password, Document Security generates a SAML authentication assertion. SAML authentication assertions are embedded in the SOAP header and returned to the client.

Subsequent messages sent to Document Security have the SAML assertion in the message header in accordance with the WS-Security standard.

NOTE: Although SAML assertions are used internally to provide session management, Document Security does not support third-party SAML assertions.

Logging in through Acrobat and other client applications

When Document Security authenticates a user through Acrobat or another client application, such as Microsoft Office, the server returns the SAML authentication assertion to the client application.

After logging in through the client application, a SAML assertion provides SSO for accessing the web application. If the client application opens the web application, users are authenticated with the assertion and are not prompted for their user name and password.

Role-based access control

Document Security uses a role-based model to control access to the web application features. Roles also determine whether users can protect documents with policies through a supported client application. You associate users and groups with roles through the User Management web pages. The role information is stored in the User Management database. For information about the roles used by Document Security, see About Document Security users in the *administrationconsole Help*. For information about assigning roles, see Managing Roles in the *administrationconsole Help*.

Document confidentiality

Document Security uses several technologies to protect documents and to provide access to them. In general, Document Security uses a symmetric cryptographic key system for encryption. Client applications such as Acrobat perform document encryption. Documents are never sent to the Document Security server when they are secured using client applications. However, they are sent to the server when secured using the Document Security Service API.

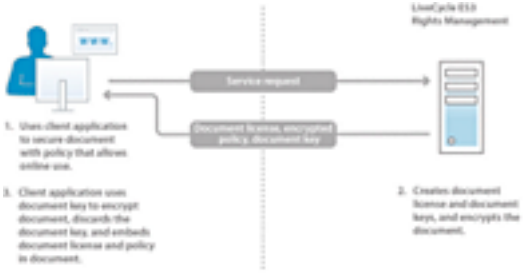
The method used to protect documents depends on whether the policy requires users to access documents while online or whether the policy enables offline use. (See *Policy-protectingdocumentsforonlineuse* and *Policy-protectingdocumentsforofflineuse*.)

When you apply a policy to a PDF document, the information that the document contains, including files that you save in the document, is protected by the confidentiality settings specified in the policy.

NOTE: Document confidentiality settings applied through a policy replace settings applied to the PDF document in Acrobat by using the password or certificate security options. (See Acrobat Help.)

Policy-protecting documents for online use

Policies can be designed so that users must be logged in to Document Security to open protected documents. Securing documents for online use employs a straightforward process for encrypting the document and providing access only to authenticated and authorized users.



The steps in the diagram are as follows:

1) The document owner or administrator decides to secure the document from a supported client application with a policy that allows online use. Users can apply policies to documents by using any supported client application. Developers can also protect documents with policies by using the Document Security service in a process or programmatically by using the Document Security Service API.
2) Document Security creates a document license and document keys, and encrypts the policy. The document license, document key, and encrypted policy are returned to the client application.

   The document license is an XML document that identifies the protected document, the policy, and the identity of the server. The server digitally signs the license to ensure data integrity.

   The document key is a symmetric key for encrypting the document. Each protected document has an associated document key.
3) The client application uses the document key to encrypt the document, discards the document key, and embeds the document license and policy. These tasks are performed in a web page or supported client application.

If the policy specifies that document events are logged, the client software sends event information to the server for logging as soon as the user opens the document. For information about the audit log, see Monitoring events in *administration console Help*.

Accessing policy-protected documents online

To open and use policy-protected documents, the policy must grant the user access to the document. The document user also needs a valid Document Security account and the appropriate client application. For PDF documents, the user needs Acrobat or Adobe Reader. For other file types, the user needs the appropriate application for the file with the Document Security extension installed.

When a user attempts to open a policy-protected document, Acrobat, Adobe Reader, or Document Security Extension connects to Document Security to authenticate the user. Then, the user can proceed to log in. If the document usage is being audited, a notification message appears. After Document Security determines which document permissions to grant, it manages the decryption of the document. The user can then use the document according to the policy confidentiality settings.

The steps in the diagram are as follows:

1) The document user opens the document in a supported client application and authenticates with the server. This task is performed in the supported client application. The document identifier is sent to the Document Security service.
2) The Document Security service authenticates the users, checks the policy for authorization, and creates a voucher. The voucher, which contains the document key and permissions, is returned to the client application.
3) The document is decrypted with the document key, and the document key is discarded. The document can then be used according to the confidentiality settings of the policy. These tasks are performed in the supported client application.

If the policy specifies that document events are logged, the client software sends event information to the server for logging as soon as the user opens the document. For information about the audit log, see Monitoring events in *administration console Help*.

If the user saves a copy of a policy-protected document, the policy is automatically applied and enforced for the new document. Events such as attempts to open the new document are also audited and recorded for the original document.

The user can continue to use a document with the following time limits:

- Indefinitely or for the validity period specified in the policy
- Until the administrator or the person who applied the policy revokes the right to open the document or changes the policy

Policy-protecting documents for offline use

Policies can be designed so that users can open documents when they do not have a network connection to Document Security.

NOTE: Documents protected with policies that allow only online use are generally more secure than documents protected with policies that allow offline use.

The first time users access an offline document, they are prompted to enable offline access. Users are not prompted again when opening any offline documents unless they open them from a different computer.

When a user enables offline access, the forms server synchronizes data with the client that is required to open documents offline. This data includes cryptographic keys and relevant updates to policies, licenses, revocation information, and so on. This data is stored in a protected client database, called a *MicroSafe*. The MicroSafe is encrypted. It is protected by using a platform-specific data protection API, such as the DPAPI for Windows and KeyChain for Mac OS®.

Each time a user opens a protected document while online, a background synchronization process is started. The synchronization updates the MicroSafe with incremental information since the last synchronization occurred. The offline lease period specified in the policy determines how recently the user must have synchronized to open documents offline. (See Configuring offline security in *administration console Help*.)

Security standards and technology

The following table provides details about the methods that Document Security uses to implement security.

Action	Technology or method used
Creating document keys Creating Initialization Vectors (IV) for AES-128 or 256-bit encryption in CBC mode	Pseudo Random Number Generator (PRNG) generated in accordance with ANSI X9.61. Implementation used is the RSA BSafe Crypto-C (in Acrobat) or Crypto-J (in Document Security) toolkits.
Encrypting PDF documents	AES-128 (or AES 256 with Acrobat 9.0) in accordance with Federal Information Processing Standards (FIPS) Publication 197.
Creating message digests	Secure Hash Algorithm-1 (SHA-1) and Secure Hash Algorithm-2 (SHA-2) in accordance with FIPS Pub 180-2.
Validating the identity of message senders	SAML authentication assertions are bound to SOAP messages. SAML assertions are hashed using SHA-1. An HMAC-SHA-1 message authentication code is used to sign the SAML assertion.

26.4. Using the Document Security service

Creating policies

You can create any number of policies by using security settings that are appropriate for different situations and users.

Policy attributes

When creating a new policy, you set various policy attributes, such as the policy name. If you are using the AEM Forms on JEE SDK or the Document Security web pages to create the policy, you can also set a validity period for the policy. A *validity period* is the time period during which a policy-protected document is accessible to authorized recipients. If you do not set this attribute, the policy is always valid.

A validity period can be set to one of these options:

- A set number of days that the document is accessible from the time it is published
- An end date after which the document is not accessible
- A specific date range for which the document is accessible
- Always valid

You can specify only a start date, which results in the policy being valid after the start date. If you specify just an end date, the policy is valid until the end date.

You can also create a policy that applies a dynamic watermark to PDF documents. Watermarks help ensure the security of a PDF document by uniquely identifying the document and controlling copyright infringement. These dynamic watermarks are different from the static watermarks that you can apply on PDF documents from Acrobat.

You can define dynamic watermarks that consist of multiple watermark elements, each containing text or a PDF. The text that is displayed in the watermark can be any of these:

User name:

The name of the user who opened the document

User ID:

The ID of the user who opened the document

Policy name:

The name of the policy that is assigned to the document

Current date:

The date the document was opened

Custom text:

Text, such as the word *Confidential*

You can define the appearance of the text, such as font, font size, and background color.

With a PDF document as the watermark element, you can include logos or special graphic indicators. For both text and PDF watermark elements, you can specify the scaling, positioning on the document, and the dynamic range of pages where the watermark should be applied. For example, you can have the watermark appearing on all pages of the documents, regardless of the number of pages.

Policy entry

A policy entry attaches principals, which are groups and users, and permissions to a policy. A policy must have at least one policy entry. Assume, for example, that you perform these tasks:

- Create a policy entry that lets a group view a document only while online and prohibits recipients from copying it
- Attach the policy entry to the policy
- Use the policy to secure a document

As a result of these actions, recipients can view the document only online and cannot copy it. The document remains secure until security is removed from it.

Methods for creating policies

- Users and administrators can use the Document Security web pages. (See “Creating and editing policies” in *Document Security Help*or *administration help.*)

- Use Workbench to create a policy from an existing policy. When you create a policy in Workbench, there are fewer properties to configure. (See WorkbenchHelp.)
- Use the Document Security Service API. (See Programming with AEM Forms.)

Modifying policies

You can modify a policy when business requirements change and the policy no longer reflects these requirements. Instead of creating a new policy, you can simply update an existing policy.

To modify a policy, you modify the value of policy attributes. The only policy attributes that you cannot change are the attributes in the Unchangeable Advanced Settings section. In Workbench, you also cannot modify the policy name and the validity time. For example, to change the policy's offline lease period, you can modify the value of the policy's offline lease period attribute.

Changes to policies that are protecting documents are updated the next time that the policy-protected document is synchronized with the Document Security service.

Methods for modifying policies

- Users and administrators can use the Document Security web pages. (See "Creating and editing policies" in *Document Security Help*or *administration console Help.*)
- Use Workbench. (See Workbench Help.)
- Use the Document Security Service API. (See Programming with AEM Forms.)

Deleting policies

You can delete an existing policy when it is no longer required. After a policy is deleted, it cannot be used to protect documents. However, existing policy-protected documents that are using the policy are still protected.

To delete a policy, you specify the policy to delete and the policy set that the policy belongs to. The user whose settings are used to invoke AEM Forms on JEE must have permission to delete the policy; otherwise, an exception occurs. Likewise, if you attempt to delete a policy that does not exist, an exception occurs.

Methods for deleting policies

- Users and administrators can use the Document Security web pages. (See "Delete a policy" in *Document Security Help* or *administration console Help.*)
- Use the Document Security Service API. (See Programming with AEM Forms.)

Applying policies to documents

You can apply a policy to a document to secure the document. Applying a policy to a document restricts access to the document and applies confidentiality settings to it.

To apply a policy to a PDF document by using the AEM Forms on JEE SDK, you must reference an existing policy and specify which policy set the policy belongs to. The user account used to connect to Document Security requires access to the policy; otherwise, an exception occurs.

Only one policy at a time can be applied to a document.

Methods for applying policies

- Use Acrobat Pro to apply policies to PDF documents. (See Acrobat Help.)
- Use Document Security Extension for Microsoft Office to apply policies to Microsoft Office documents. (See AEM Document Security Help.)
- Use Workbench. (See Workbench Help.)
- Use the Document Security Service API. (See Programming with AEM Forms.)

Removing policies from documents

You can remove a policy from a policy-protected document to remove security from the document. To update a policy-protected document with a newer policy, it is more efficient to switch the policy than to remove it and add the updated policy.

Methods for removing policies

- Use Acrobat® Pro to remove policies from PDF documents. (See Acrobat Help.)
- Use Document Security Extension for Microsoft Office to remove policies from Microsoft Office documents. (See AEM Document Security Help)
- Use Workbench. (See Workbench Help.)
- Use the Document Security Service API. (See Programming with AEM Forms.)

Switching the policy applied to a document

To update a policy-protected document with a newer policy, it is more efficient to switch the policy than to remove it and add the updated policy.

When you switch a policy, the new policy is enforced as follows:

- If the document is online and closed, the change takes effect the next time the recipient synchronizes with Document Security by opening any policy-protected document online.
- If the document is online and open, the change takes effect when the user closes the document.
- If the document is offline, the change is applied the next time the user synchronizes with Document Security by opening a policy-protected document online.

NOTE: To permit anonymous access to a policy-protected document that currently does not have this access, remove the existing policy in the client application and apply a policy that permits anonymous access. If you switch the policy, users still must log in to access the document.

Methods for switching policies

- Users and administrators can use the Document Security web pages. (See “Switch a policy that is applied to a document” in *Document Security Help*or *administration console Help*.)
- Use Workbench. (See Workbench Help.)
- Use the Document Security Service API. (See Programming with AEM Forms.)

Revoking access to policy-protected documents

You can revoke access to a policy-protected document. The result of this action is that all copies of the document are no longer accessible to users. When a user attempts to open a revoked document, they can be redirected to a specified URL where a revised document can be viewed. When you revoke access to a document, the change takes effect the next time the user opens the policy-protected document online.

The ability to revoke access to a document provides additional security. For example, assume a newer version of a document is available and you no longer want anyone viewing the outdated version. In this situation, access to the older document can be revoked, and no one can view the document unless access is reinstated.

Reinstating access to documents

You can reinstate (unrevoke) access to a revoked document. The result is that all copies of the revoked document are accessible to users.

Methods for revoking access to documents

- Users and administrators can use the Document Security web pages. (See “Revoking and reinstating access to documents” in *Document Security Help* or *administration console Help*.)
- Use Workbench. (See Workbench Help.)
- Use the Document Security Service API. (See Programming with AEM Forms.)

Monitoring events

The Document Security service can track specific actions related to the policy-protected document as they occur. This tracking happens when the auditing capability is enabled and the policy used to protect a document has auditing enabled. Events include activities such as applying a policy to a document and opening a policy-protected document.

Events fall into one of the following categories:

- Administrator events are actions related to an administrator, such as creating a new administrator account.
- Document events are actions related to a document, such as closing a policy-protected document.
- Policy events are actions related to a policy, such as creating a new policy.
- Service events are actions related to the Document Security service, such as synchronizing with the user directory.

You can use the Document Security Service API to search for specific events. You can also use the Document Security web pages to search and view audited events. Users can view audited events for their policy-protected documents and for protected documents that they receive and use. Administrators can view audited events that are related to all policy-protected documents and users. Administrators can also track other types of events, including user, document, policy, and system events.

Events that are performed on a copy of a policy-protected document are also tracked as events with the original protected document.

A failed event is recorded if an unauthorized user attempts to view a document or log in using an incorrect user name or password.

Policies can allow anonymous user access. If the administrator later turns off anonymous access, anonymous access will fail for documents protected with the policy, and the event will not be logged.

Methods for monitoring events

- Users and administrators can use the Document Security web pages. (See “Monitoring events” in *Document Security Help*or *administration help.*)
- Use the Document Security Service API. (See Programming with AEM Forms.)

26.5. Considerations for the Document Security service

Order of operations

Any combination of encrypting, certifying, and applying usage rights to the same document must occur in the following order. These services must be invoked within a short-lived process:

1) Apply encryption (Encryption service) or apply a policy (Document Security service) to a document before you digitally sign the document (Signature service). A digital signature records the state of the file at the time of signing. Encrypting the document or applying a policy after you apply a signature changes the bytes in the file, causing the signature to appear invalid.
2) Certify a PDF document (Signature service) before you set usage rights (Acrobat Reader DC extensions service). If you certify a document after you apply usage rights, it invalidates the usage rights signature, therefore removing the usage rights from the document.
3) Digitally sign a PDF document (Signature service) after you set usage rights. Signing a PDF document after applying usage rights does not invalidate the usage rights signature.

In addition, you cannot encrypt a PDF document and apply a policy to the same PDF document. Likewise, you cannot apply a policy to an encrypted PDF document.

27. Set Value Service

The Set Value service sets the value of one or more data items in the process data model. For example, you can set the value of a process variable, or you can set the value of a form field.

27.1. Using the Set Value service

You can interact with the Set Value service by developing a process in Workbench that uses the service.

You can use XPath expressions to specify the data item and the value to set it to. For every data item that you want to set the value of, you create a mapping. Variable data is persisted and available through the entire process. For information about XPath, see theXMLPath Language (XPath) specification.

For information about developing processes that use this service, see Workbench Help.

28. Signature Service

The Signature service lets you work with digital signatures and documents on the forms server. You can use this service from processes that are created in Workbench or Java or web service clients that are created using the AEM Forms on JEE SDK. For example, the Signature service is typically used in the following situations:

- The forms server certifies a form before it is sent to a user to open by using Acrobat or Adobe Reader.
- The forms server validates a signature that was added to a form by using Acrobat or Adobe Reader.
- The forms server signs a form on behalf of a public notary.

The Signature service accesses certificates and credentials that are stored in the trust store. (See Managing certificates and credentials in *administrationconsole Help*.

RELATED LINKS:

Using the Signature service

Best practices

28.1. About digital signatures

Digital signatures are blocks of information that are added to electronic documents. You can use them to authenticate the identity of the signer and verify the integrity of the document or part of the document:

- Signatures contain information that lets you determine the owner of the signature. This information is useful for verifying the identity of the originator of the document.
- When a document is digitally signed, the signature can be used to determine whether the document has changed since it was signed.

Credentials are used to create digital signatures. Certificates are used to validate digital signatures and their owners.

Digital signatures and the Signature service

The Signature service supports PDF signatures and XML signatures:

- The Signature service can create PDF signatures for all PDF documents, such as PDF documents created by using Designer or Acrobat.
- The Signature service can validate XML signatures for XFA-based PDF documents, such as PDF documents that are created by using Designer or by the Forms service.

PDF signatures can be used for purposes beyond the basic authentication of signer's identity and validating document integrity:

Approval Signatures:

Used for approving document content. For example, a user fills a form and then signs the form to approve the form data.

Certifying signatures:

Used for attesting to the document contents and specifying the types of changes that are permitted for the document to remain certified. For example, a government agency creates a form with signature fields. The agency certifies the document, allowing users to change only form fields and to sign the document. Users can fill the form and sign the document. However, if users remove pages or add comments, the document does not retain its certified status.

Certifying signatures are also known as Modify Detection Prevention (MDP) signatures.

28.2. About signature fields

When a form is digitally signed, the signature is added to a signature field. Forms must include a signature field before they can be signed.

Multiple signature fields can be added to a single form. Each signature field can be associated with a set of fields on the form. After the signature is added, the associated fields are locked. These types of signatures are known as *MDP+ signatures*. To use this feature with PDF forms and XML forms, users must use Acrobat or Adobe Reader 8.0 and later to open the forms.

Seed value dictionaries can be added to signature fields to configure how the field is used when the document is signed. For example, a list of signing reasons can be provided, or the hashing algorithms that can be used for creating the document digest can be specified.

You can add signature fields at design time or at run time:

- Use Designer to add signature fields at design time. (See Designer Help.)
- Use the Signature service to add signature fields at run time. (See *Adding,modifying,andremoving-signaturefields*.)

28.3. About the Signature service and form types

AEM Forms on JEE supports several types of PDF forms. Although Acrobat or Adobe Reader users notice no apparent difference between the form types, the way the PDF form is constructed can be different. For example, forms can be rendered to PDF by the Forms service on the forms server or by Acrobat or Adobe Reader.

PDF forms that do not require rendering can be used with the Signature service in any situation. However, PDF forms that require rendering can be problematic for digital signatures, depending on how they are used.

When a PDF form that is digitally signed is rendered, the signature on the form is invalidated. For example, a user opens a dynamic PDF form in Acrobat, digitally signs it, and saves it. Then the user sends

the file to a colleague in an email message. When the colleague opens the form, Acrobat renders the form to PDF, which invalidates the digital signature.

To use the Signature service, identify the type of form you are using.

Acrobat PDF form:

PDF forms that are created using Acrobat (or a similar tool). These forms do not require rendering after they are created.

Adobe PDF form:

PDF forms that are created by using Designer. These files are saved as static or dynamic PDF forms:

- The content of static PDF forms, except for field values, does not change. When the file is opened, Acrobat or Adobe Reader use information in the file to render the PDF form. When the file is saved for the first time, the PDF form is stored in the file. The next time the form is opened, it is not rerendered.
- The content of dynamic PDF forms can change according to user input. For example, table rows or subforms can be added as required. The PDF form is always rendered when it is opened by using Acrobat or Adobe Reader.

For more information about the static and dynamic Adobe PDF forms, see Using Designer > Working with Form Designs > Guidelines for creating PDF forms in Designer Help.

Adobe XML form:

XDP files that are created by using Designer. Adobe XML forms are prepared for opening in Acrobat or Adobe Reader by using the Forms service. The Forms service can be configured so that the PDF form is rendered by any of these agents:

- Forms service (on the forms server) before being sent to the client
- Acrobat
- Adobe Reader.

Non-interactive PDF forms:

PDF forms that users can view electronically or print. For example, files that are converted to PDF from a different file format are non-interactive. These forms do not require rendering after they are created.

For information about design requirements for forms that are used in AEM Forms on JEE Workspace, see *RequirementsforformdesignandWorkspace*.

NOTE: The Flex Worksapce is deprecated for AEM Forms.

28.4. About digital signature technology

Public key cryptography

Digital signatures are based on public-key cryptography (or asymmetric cryptography), which involves using public/private key pairs for encrypting and decrypting text:

- The private key is used to encrypt text and documents. Private keys are kept safe.
- The corresponding public key is used to decrypt the text that is encrypted by the private key. The public key can decrypt only the text that is encrypted with the associated private key. Public keys are distributed, sometimes widely.

For example, Tony Blue uses his private key to encrypt email messages before sending them to recipients. The recipients require the public key to decrypt the messages and read them. Tony must provide the recipients with the public key before they can read his email messages.

Digital certificates

Digital certificates can be used to verify the authenticity of digital signatures. Digital certificates bind a public key with a person's identity:

- Certificates can be issued by certificate authorities (CA), a trusted third party. CAs verify the identities of the people who they issue certificates to. If you trust the CA, you trust the certificates they issue.
- Certificates can also be self-signed. Self-signed certificates are typically generated by the certificate owner. Certificates are useful when you are certain that you can trust the owner.

CAs publish certificate revocation lists (CRL) that contain the serial numbers of the certificates that are no longer valid. CRLs have expiry dates and are typically updated periodically.

Similar to using CRLs, Online Certificate Status Protocol (OCSP) is used for obtaining the status of X.509 certificates. OCSP enables certificate status to be updated and obtained more quickly than CRL systems.

CAs can delegate the authority to issue certificates to lower-level CAs. The result can be a hierarchy of CAs. A certificate chain indicates the path in the hierarchy from a lower-level CA to the root CA. Certificates that are issued by lower-level CAs include the certificate chain. The authenticity of each CA in the chain can be verified.

Digital credentials

Credentials are used to digitally sign documents. A credential contains a user's private key and other identifying information, such as an alias. A password is required to access the contents of the credential. Different standards define the content of a credential and the format. The following standards are two examples:

- Personal Information Exchange Syntax Standard (PKCS #12) defines a file format for storing the private key and the corresponding digital certificate.

- Cryptographic Token Interface (PKCS #11) defines an interface for retrieving credentials that are stored in hardware.

Digital Signatures

Digital signatures are an encrypted digest of the document that is signed. The digest and the signer's certificate are used to validate the integrity of the document.

When a document is digitally signed, a digest of the document contents is created by using a hashing algorithm. The digest is unique for the document, and the document cannot be reconstructed by using the digest. The digest is encrypted by using the signer's private key to create the signature.

The signature and the certificate that corresponds with the private key that is used to create the signature are typically bundled with the document.

Signatures can include timestamps. Time Stamp Protocol (TSP) is used to establish the time at which a digital signature is created. This information is useful for verifying that a digital signature was created before the associated certificate was revoked. A Time Stamp Authority (TSA) provides services for obtaining and verifying timestamp information.

Validating document integrity

To validate the signature, the public key in the certificate is used to decrypt the digest. The digest is then recalculated and compared with the decrypted digest. If the digests are identical, the document has not been altered.

28.5. Integrating with a security infrastructure

The Signature service accesses certificates, credentials, and revocation lists that are stored in Trust Store Management. It can also use Trust Store Management to access credentials that are stored in Hardware Security Module (HSM) devices. (See Managing HSM credentials in *administration console Help*.)

The Signature service also supports communicating with external resources for retrieving certificates and validating signatures:

- LDAP/LDAPs and HTTP/HTTPs queries for retrieving certificates for chain validation.
- Connecting to TSAs using HTTP and HTTPs.
- Retrieving CRLs using HTTP/HTTPs and LDAP/LDAPs. The Signature service also supports offline CRLs that are stored using Trust Store Management.
- Connecting to OCSP servers.
- Integrating with external service providers for retrieving credentials and verifying certificates.

28.6. Supported technologies and standards

The following table provides a summary of the technologies and industry standards that Digital Signatures supports.

Item	Supported technology or standards
One-way hash (for creating document digests)	SHA-1, SHA-256, SHA-384, and SHA-512 MD5 RIPEMD160
Digital signatures	PKCS #1 and #7 RSA (up to 4096 bit) DSA (up to 4096 bit) XML signatures Seed values (enforcement of certificate usage criteria) Time stamping (using Time Stamp Providers)
Certificate validity	Certificate Revocation Lists (CRL) Online Certificate Status Protocol (OCSP) RFC 3280 compliant path validation

The Signature service enforces Federal Information Processing Standard (FIPS) compliance and uses the RSA BSAFE libraries.

28.7. HSM-based Signatures on 64-bit Windows Computers

AEM Forms on JEE uses the PKCS#11 JCE providers to support PKCS#11-compliant HSM-based signing support.

NOTE: *On 64-bit Windows platform, the 64-bit JRE from Sun and IBM do not ship with the above mentioned PKCS#11 JCE providers.*

You can take advantage of HSM-based signing support by using the web services based IPC/RPC mechanism. This method has an added advantage wherein the forms server can use HSM installed on a remote machine. To use this functionality, install the web service on the remote machine where HSM is installed. See ConfiguringHSM support using Sun JDK onWindows 64-bit platform (cpsid_80835) for more information. Unlike the CORBA-based IPC mechanism, this mechanism does not support online creation of HSM profiles or status checks. However, there are two ways to create HSM profiles and perform status checks:

- Create a AEM Forms on JEE client credential by passing it the Signer's Certificate. Follow the steps mentioned in ConfiguringHSM support usingSun JDK on Windows 64-bit platform (cpsid_80835). The web service location is passed in as a Credential property. Offline HSM profiles created either using certificate DER or certificate SHA-1 hex is also supported. However, if you have upgraded from an earlier version of AEM Forms on JEE, make client changes as the credential carries certificate and web service information.

- Web Service location is specified in the administration console for Signatures service. Here the client only carries the alias of the HSM profile in the Trust Store. You can use this option seamlessly without any client changes, even if you have upgraded from an earlier version of AEM Forms on JEE. This option does not support HSM profiles created using certificate SHA-1. ***NOTE:*** *If the web service is configured, then the BMC route is not followed.*

28.8. Using the Signature service

For information about developing processes that use this service, see Workbench Help. For information about developing client applications that programmatically interact with this service, see Programming with AEM Forms.

You can use the Applications and Services pages on Administration Console to configure default properties for this service. (See Signature service settings in *administration console Help*.)

Signing and certifying documents

You can use the Signature service to sign and certify PDF documents by using any credential that the service can access. When signing or certifying, specify the signature field to use.

The following limitations apply to dynamic Adobe PDF forms when used with the Signature service:

- You cannot sign a visible or an invisible signature field.
- You can certify invisible signature fields.
- You can certify visible signature fields only if the Signature service is configured to process documents with Acrobat 9 compatibility. The form can only be viewed using Acrobat or Adobe Reader 9.

NOTE: For all types of forms, Acrobat or Adobe Reader users can delete signatures that the Signature service added.

When signing or certifying forms, the following information can be specified:

Credential:

The credential that contains the private key to use to create the digital signature.

Document MDP permissions:

When certifying, the changes that users can perform on the document without invalidating the certification.

Revocation information:

Whether to embed revocation information in the signature to use for validating the signer's certificate. The information enables OCSP-checking and CRL-checking.

Time stamp information:

Whether to create a timestamp for the signature and the information required to perform the timestamp transaction with the timestamp provider.

Appearance:

Properties that affect the appearance of the signature when it is viewed using Acrobat or Adobe Reader. These properties can be the reason for signing, the contact information of the signer, a legal attestation, and the icons to use.

Validating document integrity and authenticity

You can use the Signature service to validate signatures that are added to PDF forms. To validate signatures, the certificate can be checked for revocation, the timestamp of the signature can be checked, and the document digest is verified. You can validate signatures individually or validate all the signatures on a PDF document.

The following limitations apply to validating digital signatures by using the Signature service:

- The Signature service cannot accurately validate signatures on dynamic Adobe PDF forms.
- The Signature service cannot ensure that field-locking rules for signature fields (MDP+ rules) are enforced for Adobe PDF forms and Adobe XML forms.

When validating signatures, the following information can be specified:

Signature field:

The name of the signature field that holds the signature to verify.

Revocation checking:

Whether to check that the signer's certificate is revoked. You can specify information to enable OCSP and CRL types of checking.

Time stamp checking:

How to verify the timestamp of the signature.

Path validation:

Information that enables the verification of the certificates in the certificate chain that the signer's certificate includes.

The validity status messages displayed depend on whether the Process Documents With Acrobat 9 Compatibility option is selected for Signature service. (See Signature service settings in *administration console Help*.)

The following table describes the situations that cause the different signature-validity states when the option is selected.

Values	Signature status
DynamicFormSignatureUnknown DocumentSignatureUnknown	Status Unknown The integrity of the document or dynamic PDF form has not been verified.

Values	Signature status
CertifiedDynamicFormSignatureTamper SignedDynamicFormSignatureTamper CertifiedDocumentSignatureTamper SignedDocumentSignatureTamper	Tamper The document or dynamic form has been altered or corrupted since the signature was applied.
SignatureFormatError	Invalid The signature is invalid because its formatting or the information it contains has errors.
DynamicFormSigNoChanges DocumentSigNoChanges	Signed with no changes The document or dynamic form has not been modified since the signature was applied.
DynamicFormCertificationSigNoChanges DocumentCertificationSigNoChanges	Certified with no changes The document or dynamic form has not been modified since it was certified.
DocSigWithChanges	Signed with changes The revision of the document that this signature covered has not been changed; however, subsequent changes were made to the document.
CertifiedDocSigWithChanges	Signed with allowed changes The document has been changed since the signature was applied. However, the changes are permitted by the document certifying party and do not invalidate the signature.
CertificationSignWithChanges	Certified with changes The document has been changed since it was certified. However, the changes are permitted by the document certifying party and do not invalidate the signature

The following table describes the situations that cause the different signature-validity states when the option is not selected.

Value	Signature status
Invalid	Signature Invalid The revision of the document that is covered by the signature has been altered.

Value	Signature status
Unknown	Status Unknown Signature validation on the signed contents was not performed.
ValidAndModified	Signature valid but document modified The revision of the document that is covered by the signature was not modified; however, subsequent changes were made to the document.
ValidUnmodified	Signature valid and document unmodified The revision of the document that is covered by the signature was not modified. No subsequent changes were made to the document.

When validating signatures, you must know whether you are validating a PDF signature or an XML signature.

Removing signatures

You can use the Signature service to remove signatures from signature fields.

Retrieving signatures and signature fields

You can use the Signature service to retrieve the following items from forms:

- Information about signature fields and certifying signature fields
- Digital signatures and information about the signatures
- The revision of the PDF form as it existed when a signature field was signed

IMPORTANT: You cannot retrieve the certifying signature field from forms that are rendered on the client.

Adding, modifying, and removing signature fields

You can use the Signature service to add, modify, and remove visible and invisible signature fields from forms. When you add and modify signature fields, you can configure the following properties:

- The field name and, for visible signature fields, the location.
- The fields to lock when the signature is added.
- The signature handler that validates signatures.
- Information about the signature (for example, whether to include revocation information, a list of signing reasons that users can select from, and server URLs used for validating signatures).
- Whether the field can be used only for certifying the document.

IMPORTANT: The Signature service cannot add or modify signature fields on a dynamic Adobe PDF form.

28.9. Best practices

The following characteristics of AEM Forms on JEE and the Signature service result in limitations to the way you can use dynamic Adobe PDF forms:

- Digital signatures are invalidated when a signed form is rendered to PDF.
- The Signature service cannot enforce field-locking (MDP+) signature rules for Adobe PDF forms and Adobe XML forms.

Generally, you must decide whether the use of dynamic PDF forms or the use of digital signatures on the server is more important for your solution:

- If you need to use features of the Signature service that do not support dynamic Adobe PDF forms, use a different type of form and ensure that no rendering occurs in Acrobat or Adobe Reader. (See *Ensuringthatnorenderingoccursaftersigning*.)
- If you need to use dynamic PDF forms, you can convert the form to a non-interactive form before using the features of the Signature service on the form. (See *Convertingtonon-interactiveform*.)
- Before you use the form with the Signature service, ensure that the form is not a dynamic Adobe PDF form. (See *Checkingtheformtype*.)

Also, to use digital signatures on forms that users open in Workspace, your form must conform to specific design criteria. (See XREF).

Order of operations

Any combination of encrypting, certifying, and applying usage rights to the same document must occur in the following order. These services must be invoked within a short-lived process:

1) Apply encryption (Encryption service) or apply a policy (Document Security service) to a document before you digitally sign the document (Signature service). A digital signature records the state of the file at the time of signing. Encrypting the document or applying a policy after you apply a signature changes the bytes in the file, causing the signature to appear invalid.
2) Certify a PDF document (Signature service) before you set usage rights (Acrobat Reader DC extensions service). If you certify a document after you apply usage rights, it invalidates the usage rights signature, therefore removing the usage rights from the document.
3) Digitally sign a PDF document (Signature service) after you set usage rights. Signing a PDF document after applying usage rights does not invalidate the usage rights signature.

Ensuring that no rendering occurs after signing

When a form is rendered to PDF, any digital signatures that it contains are invalidated. Ensure that PDF forms are not rendered after they are digitally signed. You can prevent rendering when you use static Adobe PDF forms or Adobe XML Forms.

Static Adobe PDF forms

Use Designer to create static Adobe PDF forms so that rendering occurs only the first time they are opened in Acrobat or Adobe Reader.

If you are using the Forms service to merge data with the form, prevent the service from converting the form to a dynamic PDF form. Use Designer to configure the following form properties:

Target Version:

Specify Acrobat And Adobe Reader 8.0 or Later

PDF Render Format:

Specify Static PDF Form

For information about setting form properties, see Designer Help.

If you are using the form with the User service in a process, use a Document Form variable to store the form data and the form data. Configure the Document Form variable so that the render service is called only once. (See Workbench Help.)

Adobe XML Forms

Adobe XML forms (XDP files) can be rendered to PDF by using the Forms service. To use the Signature service with Adobe XML forms, the Forms service must be configured so that rendering occurs on the server, and not on the client. When rendering occurs on the server, the rendered PDF form is embedded in the XDP file and is not rendered again.

Converting to non-interactive form

To use dynamic Adobe PDF forms and the Signature service, convert the forms to non-interactive PDF forms.

This scenario typically involves the use of dynamic Adobe PDF forms for gathering data. After the data-gathering activities are complete, the form is converted to a non-interactive PDF form that is used with the Signature service:

- Signature fields are added to the non-interactive form.
- The Signature service can add a signature to the form.
- The form can be sent to users to digitally sign, and then the Signature service can validate the signatures.

  You cannot display non-interactive forms to users in Workspace. However, you can attach non-interactive forms to the Workspace tasks by using the User service. You can also distribute the forms through email messages.

Use the Output service to convert dynamic PDF forms to non-interactive PDF forms.

NOTE: When forms that contain digital signatures are converted to non-interactive PDF forms, the digital signatures are not preserved. Only the appearance of the digital signatures is preserved.

Checking the form type

Before using the Signature service on a form, you can ensure that the form is not a dynamic Adobe PDF form. The PDF Utilities service lets you retrieve the properties of a PDF document. The results include the form type, which must not equal the value `Dynamic-XFA`.

Requirements for form design and Workspace

If you are using digital signatures on a PDF form that users open in Workspace, you must add an invisible submit button to the form instead of the Process Fields object. The invisible submit button ensures that digital signatures on the form remain valid when the form is submitted using Workspace.

When the Process Objects object is included on a form, the AEM Forms on JEE server changes the appearance of the submit button that the object provides. Digital signatures that are added before the form is submitted indicate that a change has occurred. A yellow triangle alerts the user to the change.

TIP: A submit button is a Button object with the Control Type property set to Submit. You do not have to specify a target URL. After you add the submit button, change the Presence property to Invisible.

Use Designer to add invisible submit buttons at design time. (See Designer Help.)

29. Stall Service

The Stall service is useful for preventing situational errors that you anticipate may occur. This service provides the capability to stall the branch that it belongs to.

For example, processes can use data that is provided from an external resource, such as a partner's database. An Execute Script action can be used to verify that the data is valid. If the data is not valid, a Stall action can be executed that stalls the process instance while the data in the database is corrected.

29.1. Using the Stall service

You can interact with the Stall service by developing a process in Workbench that uses the service.

If you are aware of a possible situational error in your process, you can add a route that leads to the Stall service. You add a condition to the route that checks whether the situational error occurred. When the situation occurs, the branch is stalled so that you can fix the error and restart the process using administration console.

NOTE: When used in transactional branches or short-lived processes, the execute operation throws an exception instead of stalling the branch.

For information about developing processes that use this service, see Workbench Help.

30. User Service

The User service enables processes to create and assign tasks to Workspace users, or to send tasks to users in email messages. You can also assign tasks programmatically by using the Task Manager Service API. After a task is created, you can interact with the task by using processes, the Task Manager Service API, and Administration Console.

This section includes the following topics:

RELATED LINKS:

Using the User service

Interacting with tasks

30.1. Using the User service

The User service provides the Assign Task operation that processes use to create tasks and assign them to users either in Workspace or by using email. You can also use the Assign Task operation to configure task features that are exposed to Workspace users.

Assigning tasks

Assign tasks to users or groups to add the task to their To Do list in Workspace. You can assign tasks to any user or group that exists in User Management:

- You can assign a task to the same user or group for every instance of the process that is created. This strategy can simplify testing during development. For example, if you assign all tasks to the same person, you must log in to Workspace only once to verify task creation.
- You can assign tasks to a user or group identification that is stored in a process variable. This method is more realistic for production environments because the user or group can change, depending on the process instance. Prior to assigning the task, the user or group identification is obtained and stored.

Methods for assigning tasks

- Use Workbench. (See Workbench Help.)
- Use the Task Manager Service APIs. (See Programming with AEM Forms.)

Configuring task features

When you create tasks, you can configure features that affect the user experience:

Forms and form data:

Specify which form is presented to Workspace users and the form data to include in the form. Also specify where to save the form data that is submitted when the task is completed.

Instructions:

Provide instructions to Workspace users that describe how to complete the task.

Task delegation and consultation:

Specify whether users can delegate tasks to other users or consult with other users about tasks.

Attachments and notes:

Specify whether users can add file attachments and notes to tasks. You can also add file attachments to tasks if they are relevant to the work that needs to be done.

Reminders, deadlines, and escalations:

Configure these mechanisms to ensure that the process progresses within time constraints.

Methods for configuring task features

- Use Workbench. (See Workbench Help.)

Configuring email notifications

Notification email messages can be sent to Workspace users when the following events occur:

- Users are assigned a task.
- A reminder or deadline occurs for a task that they own.
- An escalation occurs for a task that they own.

  You can configure the message subject and body. The text can include parameters for task properties that are updated at run time. Parameters enable messages to include data that changes for each task, such as the task identification.

  Default email messages can be authored by using administration console. The default messages apply to all tasks that are created for all processes. You can configure the Assign Task operation by using Workbench to override the default messages for a task, or disable notifications for the task.

  TIP: Messages that are configured using Workbench can also include XPath expressions so that you can include text-based process data in the message.

Methods for configuring email notifications

- Use Administration Console. (See Configuring notifications for users and groups in *administration console Help*.)
- Use Workbench. (See Workbench Help.)

Saving information about completed tasks

When a task is complete, you can save information about the task. Configure the Assign Task operation to store the following information in process variables:

Task identification:

The unique identification of the task that was created

User identification:

The unique identification of the user who completed the task

This information can be used later in the process if necessary. For example, you can save the user identification to assign subsequent tasks in the process to the same user.

Methods for saving task information

- Use Workbench. (See Workbench Help.)

30.2. Interacting with tasks

This section describes how to interact with tasks after they are created. Workbench, administration console, and the AEM Forms on JEE SDK provide tools for interacting with tasks.

NOTE: For information about how to interact with tasks by using Workspace, see Workbench Help.

NOTE: The Flex Worksapce is deprecated for AEM Forms.

Reacting to task events

forms workflow provides several asynchronous event types for use in processes. You can add event receives or start points to your processes to react to events when they occur.

More than 20 event types are defined for task state changes such as when a task is completed, or for other changes such as when a file is attached to the task. For example, the TaskDeadlined event type is thrown when a deadline occurs for a task. If a process uses the TaskDeadlined event as the start point, the process can be invoked when the event is thrown.

Methods for reacting to events

- Use Workbench. (See Workbench Help.)

Retrieving task information

You can retrieve tasks assigned to specific users and then retrieve information about the task, such as the task identification, status, and name of the process it belongs to. You can also retrieve a history of the task.

Methods for retrieving task information

- Use the Process Management pages in administration console. (See administration console Help.)
- Use the Task Manager Service API. (See Programming with AEM Forms.)

Intervening in task status

You can intervene in the normal progression of a process to correct problems that unpredictable circumstances cause:

- Assign tasks to various users as required. For example, you can reassign a task that is assigned to a user who changed roles in the company.
- Retry stalled tasks when the problem that caused the task to stall is fixed. For example, in an Assign Task operation, the XPath expression that assigns the task to a user included a syntax error. The error caused all tasks to stall. The error is corrected, and the tasks are retried.
- Terminate tasks that are no longer needed.

Methods for intervening in task status

- Use the Process Management pages in administration console. (See *administration console Help*.)
- Use the Task Manager Service API. (See Programming with AEM Forms.)

31. Variable Logger Service

The Variable Logger service enables processes to send messages about variable values to the system log or to log files on the file system of the forms server. When a process stalls and is not functioning as expected, this situation may be related to process variables that are not set correctly. The Variable Logger service provides the capability to track the process variables and isolate the issue causing the failure.

31.1. Using the Variable Logger service

You can interact with the Variable Logger service by developing a process in Workbench that uses the service. When using this service in a process, you can specify the following options:

- Whether to output log messages about process variables to system resources or save them to a file.
- The type of information that is logged in the case of system logging.
- The file name and path of the log file or an XPath expression to a process variable that contains the file name and path when logging to a file. You can specify whether to overwrite the log file if it already exists, create a new log file, or append to an existing log file.

  For information about developing processes that use this service, see Workbench Help.

32. Wait Point Service

The Wait Point service lets you delay the progression of a process at a step in the process.

32.1. Using the Wait Point service

You can interact with the Wait Point service by developing a process in Workbench that uses the service. When you use this service in a process to delay the execution of the next operation in the process, specify the amount of time to wait by using either calendar days or business calendar days:

Calendar days:

When you use calendar days, provide values for the number of days, hours, minutes, and seconds to wait.

Business calendar days:

When you use business calendar days, specify the name of the business calendar to use and the number of business days to wait. *Business calendars* define business and non-business days (for example, statutory holidays, weekends, and company shutdown days) for your organization. When using business calendars, AEM Forms on JEE skips non-business days when calculating the amount of time to wait.

For information about developing processes that use this service, see Workbench Help.

33. Web Service Service

The Web Service service enables processes to invoke web service operations. For example, an organization may want to integrate a process to store and retrieve information such as contact and account details by invoking a service provider's exposed web services. The Web Service service invokes a specified web service and passes through values for each of its parameters. It then saves the return values from the operation into a designated variable within a process.

The Web Service service interacts with web services by sending and receiving SOAP messages. The service also supports sending MIME, MTOM, and SwaRef attachments with SOAP messages by using the WS-Attachment protocol. The Web Service service interactions are compatible with SAP systems and .NET web services.

33.1. Using the Web Service service

You can interact with the Web Service service by developing a process in Workbench that uses the service. You can accomplish the following tasks by using this service:

- Create the SOAP message to send to the web service for invoking a web service operation. After you provide the URL to the web service definition, you can select the web service operation to invoke. Based on the operation that you select, a template of the SOAP request message is generated. You then insert values into the message as required. The Web Service service supports the Table data type, which SAP web services can require. An SAP table is similar to a database table composed of columns, where each row in the table represents a record.
- Test your invocation request by sending a test message and displaying the response message that the web service sends.
- Invoke a web service operation and save the response as process data, including attachments.

  For information about developing processes that use this service, see Workbench Help.

34. XMP Utilities Service

PDF documents contain metadata, which is information about the document (as distinguished from the contents of the document, such as text and graphics). The Adobe Extensible Metadata Platform (XMP) is a standard for handling document metadata.

The XMP Utilities service can retrieve and save XMP metadata from PDF documents and import XMP metadata into PDF documents.

34.1. About XMP metadata

XMP provides a standard format for creating, processing, and exchanging metadata for a wide variety of applications. XMP provides a model by which metadata is represented. XMP metadata is encoded as XML-formatted text that uses the W3C standard Resource Description Language (RDF).

In XMP, metadata consists of a set of properties that are associated with a document. Metadata includes properties such as the author, title, and modification date of a document.

Properties can sometimes be associated with components of a document, but the XMP Utilities service does not provide the ability to manipulate component metadata.

Properties have names and values:

- Names must be legal XML names.
- Values may be simple values, such as numbers and strings, or arrays (also called *containers*). All values are actually represented as Unicode strings.

34.2. About metadata in PDF documents

In a PDF file, metadata can be stored in two places:

- In the Info dictionary of the file trailer dictionary. This dictionary contains information about the file, such as title, author, and creation date. This information is stored as PDF objects such as strings and dates, not in XML format.

  The information in this dictionary is visible to Acrobat and Adobe Reader users through the document properties. Users can set some of the properties, such as Title, Author, Subject, and Keywords. Users can also add custom properties with a unique name and value.
- In the Metadata dictionary of the document catalog. This dictionary contains metadata that is associated with the entire document. This information is represented as XMP metadata.

  NOTE: Individual streams in a document, such as images, may also have metadata entries that contain associated XMP metadata. However, the XMP Utilities service does not provide the ability to manipulate such component-level metadata.

All metadata in the Info dictionary is also represented in the Metadata dictionary in the form of XMP metadata properties. The standard properties, such as Title and Author, are represented in XMP as properties from the PDF schema.

When the XMP Utilities service reads metadata from a PDF file, it resolves inconsistencies between values in the Info dictionary and those in the XMP metadata:

- If the Info dictionary is newer, the Info dictionary properties are used to update the XMP metadata.
- If the XMP metadata is newer, the XMP properties are used to update the Info dictionary.
- Properties in the Info dictionary that are not listed in "Document Information Dictionary" in the *PDF Reference* are mapped to the `pdfx` namespace ("http://ns.adobe.com/pdfx/1.3/"). This mapping is used when copying properties between the repositories in the situations described in the first two points.

When a PDF document is saved, some metadata properties are automatically updated, specifically, `xmp:ModifyDate`, `xmp:MetadataDate`, `xapMM:InstanceID` and, if missing, `xapMM:DocumentID`. If you attempt to modify these properties, values you specify will be overridden.

34.3. Using the XMP Utilities service

You can accomplish the following tasks by using this service:

Export Metadata:

Exports the metadata from a specified PDF document that you can save as process data. The PDF document you specify can be stored as process data or specified directly as a file from the file system.

Import Metadata:

Imports metadata from process data and replaces the existing metadata in a specified PDF document.

Export XMP:

Available only in Workbench. Exports the metadata as XMP data from a specified PDF document. You can save the metadata as a PDF document to process data or a file to be reused.

Import XMP:

Available only in Workbench. Imports metadata from a document value and replaces the existing metadata in a specified PDF document.

For information about developing processes that use this service, see Workbench Help. For information about developing client applications that programmatically interact with this service, see Programming with AEM Forms.

35. XSLT Transformation Service

The XSLT Transformation service enables processes to apply Extensible Stylesheet Language Transformations (XSLT) on XML documents.

35.1. Using the XSLT Transformation service

You can interact with the XSLT Transformation service by developing a process in Workbench that uses the service. You can accomplish the following tasks by using this service:

- Configure the service with the default Java class to use for performing the XSLT transformation. The service operation properties can override the value that you provide in the service configuration.
- Transform an XML document by using an XSLT script that is stored as process data. The XML document is also stored as process data.
- Transform an XML document by using an XSLT script that is referenced using a URL. The XML document that is transformed is stored as process data.
- Test the transformation and view the result.
- Save the resultant XML document as process data.

  For information about developing processes that use this service, see Workbench Help.

  You can use the Applications and Services pages in administration console to configure default properties for this service. (See XSLT Transformation service settings in administration console Help.)