



Adobe

LiveCycle® ES Services

July 2007

Adobe® LiveCycle® ES

Version 8.0

© 2007 Adobe Systems Incorporated. All rights reserved.

Adobe® LiveCycle® ES Services for Microsoft® Windows®, Linux®, and UNIX®
Edition 1.1, July 2007

If this guide is distributed with software that includes an end user agreement, this guide, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of such license. Except as permitted by any such license, no part of this guide may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of Adobe Systems Incorporated. Please note that the content in this guide is protected under copyright law even if it is not distributed with software that includes an end-user license agreement.

The content of this guide is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Adobe Systems Incorporated. Adobe Systems Incorporated assumes no responsibility or liability for any errors or inaccuracies that may appear in the informational content contained in this guide.

Please remember that existing artwork or images that you may want to include in your project may be protected under copyright law. The unauthorized incorporation of such material into your new work could be a violation of the rights of the copyright owner. Please be sure to obtain any permission required from the copyright owner.

Any references to company names, company logos and user names in sample material or sample forms included in this documentation and/or software are for demonstration purposes only and are not intended to refer to any actual organization or persons.

Adobe, the Adobe logo, Acrobat, Distiller, Flash, Flex Builder, LiveCycle, PostScript, and Reader are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

EMC and Documentum are registered trademarks of EMC Corporation in the United States and around the world. Copyright 1994-2007 EMC Corporation, all rights reserved.

IBM and FileNet are trademarks of International Business Machines Corporation in the United States, other countries, or both.

JBoss is a trademark or registered trademark of Red Hat, Inc. in the United States and other countries.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Microsoft and Windows are either registered trademarks or a trademarks of Microsoft Corporation in the United States and/or other countries.

Sun, Java, and JavaScript are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

UNIX is a registered trademark of The Open Group in the US and other countries.

All other trademarks are the property of their respective owners.

This product contains either BISAFE and/or TIPEM software by RSA Data Security, Inc.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).

This product includes code licensed from RSA Data Security.

This product includes software developed by the JDOM Project (<http://www.jdom.org/>).

Macromedia Flash 8 video is powered by On2 TrueMotion video technology. © 1992-2005 On2 Technologies, Inc. All Rights Reserved.
<http://www.on2.com>.

This product includes software developed by the OpenSymphony Group (<http://www.opensymphony.com/>).

Portions of this code are licensed from Nellymoser(www.nellymoser.com)

MPEG Layer-3 audio compression technology licensed by Fraunhofer IIS and THOMSON Multimedia (<http://www.iis.fhg.de/amm/>).

This product includes software developed by L2FProd.com (<http://www.L2FProd.com/>)

The JBoss library is licensed under the GNU Library General Public License, a copy of which is included with this software.

The BeanShell library is licensed under the GNU Library General Public License, a copy of which is included with this software.

This product includes software developed by The Werken Company (<http://jaxen.werken.com/>).

This product includes software developed by the IronSmith Project (<http://www.ironsmith.org/>).

The OpenOffice.org library is licensed under the GNU Library General Public License, a copy of which is included with this software.

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

Notice to U.S. Government End Users. The Software and Documentation are "Commercial Items," as that term is defined at 48 C.F.R. §2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation," as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §§227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished-rights reserved under the copyright laws of the United States. Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, USA. For U.S. Government End Users, Adobe agrees to comply with all applicable equal opportunity laws including, if appropriate, the provisions of Executive Order 11246, as amended, Section 402 of the Vietnam Era Veterans Readjustment Assistance Act of 1974 (38 USC 4212), and Section 503 of the Rehabilitation Act of 1973, as amended, and the regulations at 41 CFR Parts 60-1 through 60-60, 60-250, and 60-741. The affirmative action clause and regulations contained in the preceding sentence shall be incorporated by reference.

Contents

Preface	6
What's in this guide?	6
Who should read this guide?	6
Related documentation	6
1 Introducing LiveCycle ES Services	7
How developers interact with services.....	7
How administrators interact with services.....	7
2 Assembler Service	8
How developers use the Assembler service	8
Describing results with Document Description XML (DDX).....	8
Associating documents to names in the DDX document.....	9
Source documents	9
Result documents.....	9
Document assembly	10
Document disassembly	11
Document manipulation.....	12
3 Barcoded Forms Service	14
How developers use the Barcoded Forms service	14
Development considerations	17
Encoding and decoding formats	17
User-specified character sets.....	17
4 Connector Services for ECM	18
How developers use the ECM connector services.....	18
How administrators use the ECM connector services.....	19
5 Convert PDF Service	20
How developers use the Convert PDF service	20
6 Decision Point Service	21
How developers use the Decision Point service	21
7 Distiller Service	22
How developers use the Distiller service	22
How administrators use the Distiller service	22
8 Email Service	23
How developers use the Email service	23
9 Encryption Service	24
How developers use the Encryption service	24
Development considerations	24
10 Execute Script Service	25
How developers use the Execute Script service.....	25
11 FTP Service	26
How developers use the FTP service.....	26
Development considerations	26

12 File Utilities Service	27
How developers use the File Utilities service	27
Development considerations	27
13 Form Augmenter Service	28
How developers use the Form Augmenter service.....	28
14 Form Data Integration Service	29
How developers use the Form Data Integration service.....	29
15 Forms Service	31
Development considerations	32
Creating form designs for the Forms service	33
Processing requests.....	33
Requesting a form	33
Using form design buttons	35
Submit button	35
Calculate button	38
16 Generate PDF Service	40
How developers use the Generate PDF service	40
How administrators use the Generate PDF service.....	41
17 JDBC Service.....	42
How developers use the JDBC service.....	42
Development considerations	42
18 JMS Service.....	43
How developers use the JMS service	43
Development considerations	43
19 LDAP Service	44
How developers use the LDAP service	44
Development considerations	44
20 Output Service	45
How developers use the Output service.....	45
Development considerations	45
Form data	45
Form type	46
Email support	46
Print specification	46
ZPL output.....	46
Printable areas	46
Scripts	47
Device profile (XDC file).....	47
21 PDF Utilities Service	48
How developers use the PDF Utilities service.....	48
22 Reader Extensions Service	49
How developers use the Reader Extensions service.....	49
Development considerations	50
Configuring the credential.....	50
Order of operations.....	50
Adding usage rights to interactive forms	50
Opening rights-enabled PDF documents.....	50

23 Repository Service	51
How developers use the Repository service.....	51
24 Rights Management Service	52
About policies and policy sets	52
Security methods and technology.....	53
Authentication	53
Methods of authentication	53
SAML authentication assertions.....	53
Authorization.....	54
Document confidentiality.....	55
Document protection for online use	55
Document access for online use	57
Document protection for offline use.....	58
Synchronization for offline use.....	59
Document access for offline use	59
Security standards and technology.....	60
How developers use the Rights Management service	61
25 Set Value Service	62
How developers use the Set Value service.....	62
26 Signature Service	63
About public key technology	63
About certificates and credentials	63
Integrating with an existing security infrastructure	64
How developers use the Signature service.....	64
Development considerations	65
Order of operations.....	65
Operation compatibility.....	65
27 Stall Service	67
How developers use the Stall service.....	67
28 User Service	68
How developers use the User service	68
29 Variable Logger Service	69
How developers use the Variable Logger service	69
30 Wait Point Service	70
How developers use the Wait Point service	70
31 Web Service	71
How developers use the Web Service	71
32 XMP Utilities Service.....	72
About XMP metadata	72
About metadata in PDF documents.....	72
How developers use the XMP Utilities service.....	73
33 XSLT Transformation Service	74
How developers use the XSLT service	74

Preface

This guide describes the services that developers can use to create Adobe® LiveCycle® ES (Enterprise Suite) applications.

What's in this guide?

This guide provides introductory information about services in LiveCycle ES 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.

Who should read this guide?

This guide is primarily intended for people who are designing processes in Adobe LiveCycle Workbench ES or developers who want to build client applications that programmatically interact with services. The guide also includes information of interest to administrators who manage LiveCycle ES servers and applications.

Related documentation

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

For information about	See
LiveCycle ES solution components	<i>LiveCycle ES Overview</i> at http://www.adobe.com/go/learn_lc_overview
The services licensed for use with each LiveCycle ES solution component	<i>Services for LiveCycle ES Solution Components</i> at http://www.adobe.com/go/learn_lc_services
The programming interfaces licensed for use with each LiveCycle ES solution component	<i>Programming Interfaces for LiveCycle ES Solution Components</i> at http://www.adobe.com/go/learn_lc_APIComponent
Using a service in a process map	<i>LiveCycle Workbench ES Help</i>
Using a service's API	<i>LiveCycle ES SDK Help</i> at http://www.adobe.com/go/learn_lc_programming
Administering a service	<i>Administering LiveCycle ES</i> at http://www.adobe.com/go/learn_lc_administration

1

Introducing LiveCycle ES Services

Each LiveCycle ES solution component makes use of several services to accomplish different tasks as part of a business process. This document describes the services that developers can use to create LiveCycle ES applications. Each service is licensed for use with one or more LiveCycle ES solution components in a production environment. The LiveCycle Foundation services are licensed for use with all LiveCycle ES solution components.

How developers interact with services

When a service or application is deployed within LiveCycle ES, it can be invoked by a client application using different mechanisms. From the LiveCycle 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 LiveCycle Workbench ES 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 Flex Builder that uses LiveCycle Remoting to interact with the service. Most services can be invoked using LiveCycle Remoting.

A service can also be invoked using email and Watched Folders. For more information about the different ways in which to invoke services, see "Invoking LiveCycle ES" in *LiveCycle ES SDK Help*.

How administrators interact with services

You can use the LiveCycle Administration Console to perform these tasks:

- Configure and manage users, groups, and server authentication settings using User Management.
- Create and manage invocation endpoints and deploy LiveCycle ES archive (LCA) files.
- Set up Watched Folders and email providers for non-programmatic process invocation.
- Administer solution component properties and server settings such as port numbers and log files.

The Assembler service can assemble multiple PDF documents into one PDF document or disassemble one PDF document into multiple PDF documents. The Assembler service can manipulate documents in various ways such as changing page size and rotating contents. It can insert additional content such as headers, footers, and a table of contents, as well as preserve, import or export existing content such as annotations, file attachments, and bookmarks.

Starting in LiveCycle ES 8.0, support for PDF packages is now available with the Assembler service. For more information about PDF packages, see *LiveCycle ES SDK Help*.

How developers use the Assembler service

You can interact with the Assembler service in one of several ways:

- Develop a process in Adobe LiveCycle Workbench ES that uses the Assembler service
- Develop a client application that uses the Assembler service API in Java™ or in an environment that permits you to use its exposed WSDL, such as Microsoft Visual Studio .NET
- Develop a client application in Flex™ Builder™ that uses LiveCycle Remoting to invoke the Assembler service

For information about developing processes that use this service, see *LiveCycle Workbench ES Help*. For information about developing client applications that programmatically interact with this service, see *LiveCycle ES SDK Help*.

Describing results with Document Description XML (DDX)

To specify documents that you want the Assembler service to produce, you use an XML-based language called *Document Description XML (DDX)*.

DDX is a declarative markup language whose elements represent building blocks of documents. These building blocks include PDF pages and other elements such as comments, bookmarks, and styled text.

When using the Assembler service, you create DDX documents that describe the output you want. In addition to PDF documents, DDX can describe output formats that represent data extracted from PDF documents (such as comments, text, form data, file attachments, and bookmarks) or provide information about the properties of a PDF document.

The next sections explain the association between documents with DDX expressions and show examples of typical tasks that the Assembler service can perform, as specified in the DDX documents submitted by the user.

Associating documents to names in the DDX document

When your client application invokes the Assembler service, it supplies a DDX document and a set of source documents. When the Assembler service completes processing of the DDX document, it returns a set of result documents that are also represented as objects. The set of source documents and the set of result documents associate a name with each document.

Source documents

When your client application prepares to invoke the Assembler service, it creates a data structure (a hash map) that associates a name with each of the source documents, which are represented as `Document` objects. These names must correlate with the source document names specified in the DDX document, as shown in the following illustration.

The names your client provides for source documents are logical names. A name is not tied to the underlying source of the document's contents, whether that source is a file, database, or another service.

The following illustration shows a map containing the source documents named `Doc1`, `Doc2`, and `Doc3`, each representing a `Document` object. `Doc4` is the resultant document created by the DDX segment.

Map with source documents	Corresponding DDX segment
Doc1 - Document object	<pre><PDF result="Doc4"> <PDF source="Doc1"/> <PDF source="Doc2"/> <PDF source="Doc3"/> </PDF></pre>
Doc2 - Document object	
Doc3 - Document object	

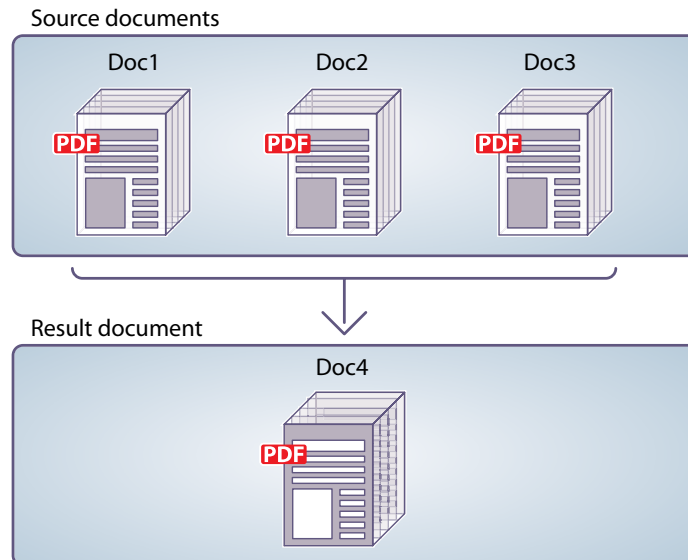
Result documents

The Assembler service returns a set of documents, which are also represented as a data structure (a map). Each entry in the map associates a name with a `Document` object. These names correspond to the result document names specified in the DDX document, as shown in the following illustration. The illustration shows a map containing the resultant document named `Doc4`, which is assembled from the corresponding DDX.

Map with source document	Corresponding DDX segment
Doc4 - Document object	<pre><PDF result="Doc4"> <PDF source="Doc1"/> <PDF source="Doc2"/> <PDF source="Doc3"/> </PDF></pre>

Document assembly

This figure shows how multiple source documents can be assembled into a single result document. Doc1, Doc2, and Doc3 are source documents that are assembled into Doc4, which is the result document.



The DDX document specifies the names of the source documents that should be used to produce the result document as well as the name of the result document.

Example: A simple DDX expression that assembles a document

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

Document assembly produces a result document that contains the following:

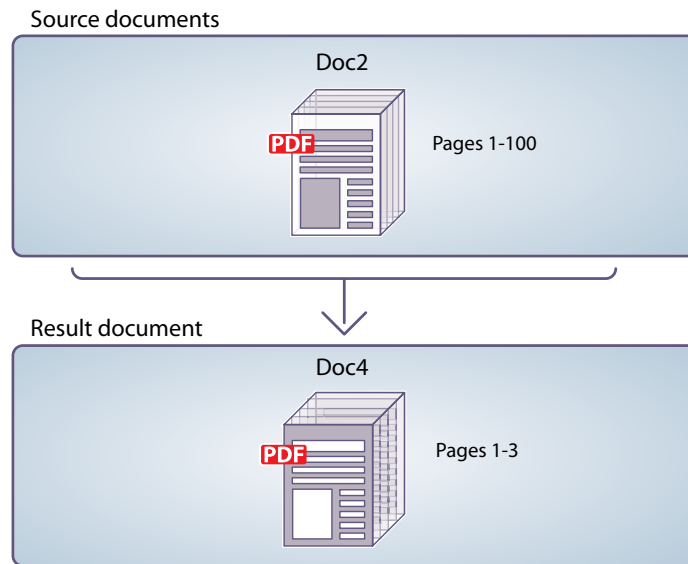
- All or part of each source document
- All or part of the bookmarks from each source document, normalized for the assembled result document
- Other characteristics adopted from the base document (Doc1), including metadata, page labels, and page size
- Optionally, the result document may include a table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

Document disassembly

The DDX document can disassemble a document by extracting pages from the source document or by breaking apart a source document based on bookmarks.

Extracting pages from a source document

In the following figure, pages 1 to 3 are extracted from the source document and placed in a new result document.



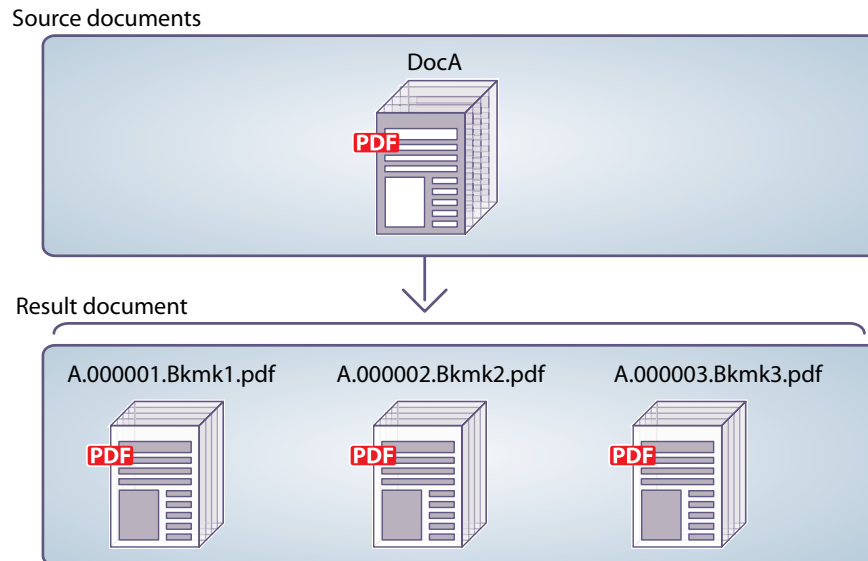
The following example shows the DDX expression that would produce this result document.

Example: A simple DDX expression that extracts pages 1 to 3 from a source document

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

Breaking apart a source document based on bookmarks

In the following figure, DocA is split into multiple result documents, where the first level-1 bookmark on a page identifies the start of a new result document.



The following example shows the DDX expression that would produce these result documents.

Example: A simple DDX expression that uses bookmarks to disassemble a source document

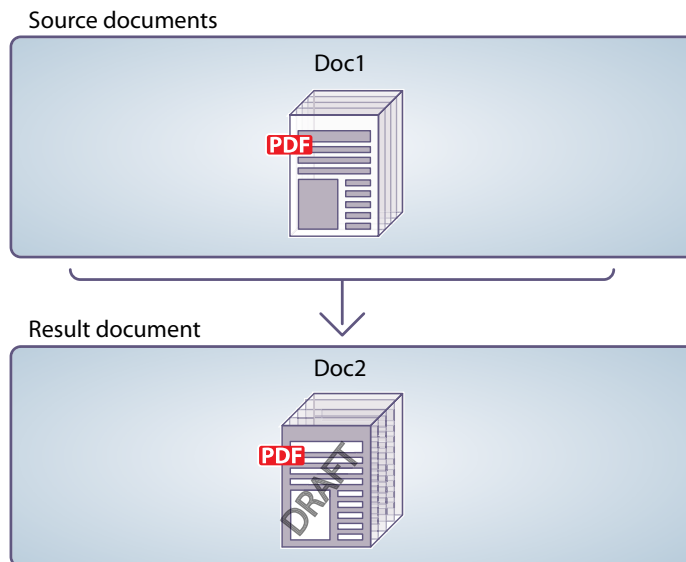
```
<PDFsFromBookmarks prefix="A">  
  <PDF source="DocA"/>  
</PDFsFromBookmarks>
```

Document manipulation

A DDX document may specify document manipulation, in conjunction with document assembly or disassembly. Manipulation may involve any combination of the following effects:

- Add or remove watermarks or backgrounds on selected pages.
- Add or remove headers and footers on selected pages.
- Create or flatten a PDF package.
- Renumber page labels. Page labels are typically used for page numbering.
- Import metadata from another source document.
- Add or remove file attachments, bookmarks, links, and comments.
- 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.

This figure shows a watermark being added to all pages in the result document.



The following example shows the DDX expression that would produce this result document.

Example: A simple DDX expression that adds a watermark to a document

```
<PDF result="Doc2">  
  <PDF source="Doc1">  
    <Watermark rotation="45">  
      <StyledText><p>  
        <b font-size="120pt">DRAFT</b></p>  
      </StyledText>  
    </Watermark>  
  </PDF>  
</PDF>
```

3

Barcoded Forms Service

The Barcoded Forms service extracts barcode data from scanned images. The service accepts a barcoded form (TIFF or PDF) as input and extracts the machine representation of the data encoded by the barcode. The barcode data may be formatted in a variety of ways, including XML, delimited string, or any custom format created with JavaScript.

The Barcoded Forms service supports the following two-dimensional symbologies:

- PDF417
- Data Matrix
- QR Code

In addition, the service supports the following one-dimensional symbologies:

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

How developers use the Barcoded Forms service


You can accomplish the following tasks using this service:

- Extract barcode data from scanned images (TIFF or PDF) and route the data to the appropriate business processes based on the form type or the data itself.
- Convert delimited data to XML (XDP or XFDF) so that routing can be performed based on the contents of the barcode. This allows data that was retrieved from barcodes to be used in other service operations that require XDP or XFDF data.

Form authors create the interactive barcoded forms using Designer ES or Acrobat Professional. You can then publish the barcoded forms to a website or distribute them by email or CD. When a user fills a barcoded form by using Adobe Reader or Acrobat, the barcode is updated automatically to encode the user-supplied form data. The user can submit the form electronically or print it to paper and submit it by mail or fax.

The typical starting point of an application that contains Barcoded Forms operations is a Watched Folder. Document scanning applications place TIFF or PDF images of scanned barcoded forms into a folder. The Watched Folder end point passes these images to the process for decoding and data handling.

Consider the following scanned TIFF image of a barcoded form, which contains two barcodes:



University of Higher Learning

Student Name Change Form

Student ID # 90210

Name as it appears on University records:

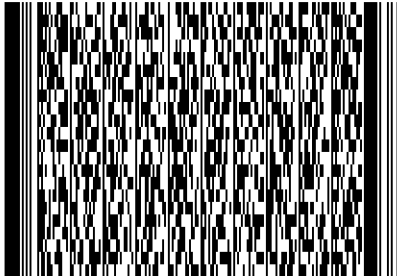
First Patti Middle Y Last Penne

Enter your new name as you would like it to appear on University records:

First Patti Middle P Last Prosciutto

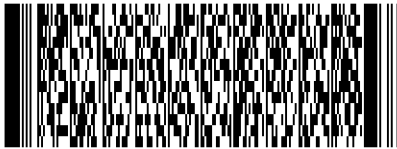
Signature

For Official Use Only - Barcodes are tab-delimited



Tab-Delimited Values Shown on Right

Field	Value
t_SID	90210
t_FirstName	Patti
t_MiddleName	Y
t_LastName	Penne
t_nFirstName	Patti
t_nMiddleName	P
t_nLastName	Prosciutto



Tab-Delimited Values Shown on Right

t_FormType

t_FormVersion

Any reference to company names, company logos, identifiers, and persons in the sample forms included in this software is for demonstration purposes only and is not intended to refer to any actual organization or individual.

The Barcoded Forms ES Decoder locates each barcode on the scanned image, decodes it, and extracts the data in the specified format. The Decoder places the barcode data (using entity encoding where required) in a content element in an XML message called the Decoder Output Data (DOD). Here is a sample of the DOD:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xb:scanned_image xmlns:xb="http://decoder.barcodeforms.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>
```

The DOD can be processed by additional LiveCycle ES components. For example, you can use the `extractToXML` operation to convert delimited data to XML (XDP or XFDF) and route the data to another service that requires XDP or XFDF data.

Development considerations

There are factors to consider when developing an application that uses barcoded forms.

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 avoid using Carriage Return as the field delimiter. Designer ES has a selection for delimited encoding that automatically generates a JavaScript script to encode barcodes with the field names on the first line and their values on the second line with tabs between each field.

When form authors use the recommended tab-delimited format, developers should use the default value of `Tab` for the field delimiter in the `extractToXML` operation when converting to XML.

User-specified character sets

The `barcode` element includes a `charEncoding` property. When form authors add barcode objects to their forms using Designer ES, 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. The `decode` operation enables you to specify the character set encoding value used in the barcode. To guarantee that all data is decoded correctly, specify the same character set as the one specified by the form author when the form was designed.

Adobe LiveCycle ES Connector for EMC Documentum and Adobe LiveCycle ES Connector for IBM® FileNet each provide two services:

A repository provider service: The Repository Provider service for EMC Documentum and the Repository Provider service for IBM FileNet provide storage and retrieval capabilities through the Repository API. Either service enables developers to access and manage assets stored in the ECM content repository at design time.

When a developer designs a form, and creates an application in Workbench ES or using the Repository API, the developer can use the assets in an ECM content repository instead of the repository provided with LiveCycle ES. Such assets may include forms, PDF files, fragments, images, XML schemas, and test data. Developers can create an application allowing LiveCycle ES services to access these assets through the Repository API, process variables, and property sheets. Form designers can access and store these assets directly in the ECM content repository within Workbench ES.

A content repository connector service: The Content Repository Connector service for EMC Documentum and Content Repository Connector service for IBM FileNet are both independent and usable as operations in a LiveCycle ES process. Workbench ES developers can use either of these services in a process to store content in 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.

Take the example of a financial institution automating an account opening process. The process must enable applicants to 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 or post. The LiveCycle ES application that the financial institution develops for this process includes content repository connector service operations to store the completed forms in a customer-defined content object type in the ECM repository and to retrieve the documentation, which can be generated from other ECM applications, for assembly from the ECM repository.

At run time, 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 Workspace ES, EMC Documentum Webtop, or IBM FileNet P8 Workplace. Or a client application can retrieve and store content as part of an automated business process.

How developers use the ECM connector services

In order to use an ECM repository provider service in Workbench ES or programmatically using the Repository API, administrators must select the appropriate ECM repository service provider in the LiveCycle Administration Console (instead of the default repository provider service provided with LiveCycle ES).

You can access an ECM content repository by using the LiveCycle Form Design perspective in Workbench ES. You will be logged in to the default Documentum repository or FileNet object store. You can also specify a Documentum or FileNet repository when you log in to Workbench ES. You must have the appropriate credentials to access the content repository. Each time you use Workbench ES, a connection to the selected content repository is made. The content repository is exposed as a hierarchical directory

structure (below a docbase or objectstore directory) in the Resources View in Workbench ES. You can share the content repository from Workbench ES with other developers.

When developing a process, you can specify the resource URL to the ECM content repository in the properties associated with a service, such as the Forms service. You can also specify the resource URL using the Forms service API.

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 used by the repository provider).
- 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.

You can also access an ECM content repository using the Repository 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.

For information about developing a process that uses each content repository connector service, see *LiveCycle Workbench ES Help*. For information about developing a client application that programmatically interacts with each ECM content repository using the Repository API, see “Working with Repositories” in *LiveCycle ES SDK Help*.

How administrators use the ECM connector services

When you install LiveCycle ES Connector for EMC Documentum or LiveCycle ES Connector for FileNet in the LiveCycle ES configuration during setup, you must specify the location of the ECM Java libraries. For more information, see the “Configuring the LiveCycle ES Connector for ECM configuration” section in the *Installing and Deploying LiveCycle ES* document for your application server.

You configure the default connection to the ECM content repository, change the repository service provider, and specify other ECM-specific default settings by using the LiveCycle Administration Console. For more information, see *LiveCycle Administration Console Help*.

The Convert PDF service converts PDF documents to PostScript® and to a number of image formats (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.

How developers use the Convert PDF service

You can accomplish the following tasks using this service:

- Convert PDF documents to PostScript. When converting to PostScript, the conversion operation allows you to specify the source document and whether to convert to PostScript level 2 or 3.
- Convert PDF documents to JPEG, JPEG 2000, PNG, and TIFF image formats. When converting to any of these image formats, the conversion operation allows you to specify the source document and an image options specification containing various preferences. Examples of such preferences include image conversion format, image resolution, and color conversion.
- Unzip an archive that contains JPEG, JPEG 2000, or PNG image formats. When converting to these image formats, the resulting image files are output to a ZIP file. This operation allows you to extract the source ZIP file and specify where to place the image files.

For information about developing processes that use this service, see *LiveCycle Workbench ES Help*. For information about developing client applications that programmatically interact with this service, see *LiveCycle ES SDK Help*.

6

Decision Point Service

The Decision Point service provides the capability to identify a point in the process where a decision is made that affects how the process progresses.

For example, a customer can fill an invoice dispute form on a corporate web site. Several routes need to be evaluated to determine the first operation to execute in a process. The dollar amount of the invoice determines whether the form is routed to a first-level manager for approval or to a credit representative for processing.

How developers use the Decision Point service

You can use this service in a process when you need multiple routes to originate at an operation but there is no single step in the business process that requires the evaluation of routes. 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 *LiveCycle Workbench ES Help*.

The Distiller® service converts PostScript, Encapsulated PostScript (EPS), and PRN files to compact, reliable, and more secure PDF files over a network. 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.

How developers use the Distiller service

When converting a PostScript, Encapsulated PostScript, or PRN file to a PDF file, the conversion operation allows you to specify several options to apply to the resulting PDF document. For example, you can specify the following options:

- Adobe PDF settings, such as high quality print, oversized pages, and press quality. When this parameter is not provided, the Distiller service will use the default PDF settings as configured using the LiveCycle Administration Console.
- Job configuration settings to apply while generating the PDF document, such as optimizing for fast web viewing. This optional parameter contains the file with settings to apply while generating the PDF document (for example, optimization for web view) and settings to apply when the PDF document is created (for example, initial view and security).
- Metadata information to apply to the generated PDF document. This optional parameter contains the file with metadata information to apply to the generated PDF document. Only UTF-8 encoded Adobe Extensible Metadata Platform (XMP) metadata is supported. For details of the format and for the specifications, please visit http://www.adobe.com/go/learn_lc_XMP.

For information about developing processes that use this service, see *LiveCycle Workbench ES Help*. For information about developing client applications that programmatically interact with this service, see *LiveCycle ES SDK Help*.

How administrators use the Distiller service

You can use the LiveCycle Administration Console to manage user access and set run-time properties that control the way this service operates.

From the LiveCycle Administration Console, you can configure this service on the LiveCycle ES PDF Generator page. You can specify default PDF settings and security settings to apply when converting to PDF.

For information about administrative tasks, see *Administering LiveCycle ES*.

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, which 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 of identifying a unique email message from several that may exist in the mailbox. 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.

How developers use the Email service

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

- Configure the Email component with default properties for connecting to an SMTP server for sending email messages, and 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 *LiveCycle Workbench ES Help*.

The Encryption service provides the ability to encrypt and decrypt documents. When a document is encrypted with a password, its contents become unreadable. Passwords are used to restrict the ability of users to open a PDF document and perform operations on it. An authorized user can decrypt the document to obtain access to the contents. The user must specify the open password before the document can be viewed within Adobe Reader or Acrobat.

PDF files are structured as a collection of objects of different types. These objects include basic types, such as numbers and strings, as well as complex types, such as dictionaries and streams. When encrypting a PDF document, only string and stream objects are encrypted. Streams are used to represent page contents while strings are used for representing text.

How developers use the Encryption service

You can accomplish the following tasks using this service:

- Encrypt a PDF document with a password. You can choose to encrypt the entire PDF document (content, metadata, and attachments), encrypt everything other than its metadata, or encrypt only the attachments.
- Remove password-based encryption from a PDF document.
- Unlock the PDF document so that other service operations can be performed. For example, after a password-encrypted PDF document is unlocked, you can use the Signature service to apply a digital signature to the document.

For information about developing processes that use this service, see *LiveCycle Workbench ES Help*. For information about developing client applications that programmatically interact with this service, see *LiveCycle ES SDK Help*.

Development considerations

If you are working with multiple services, it is important to perform service operations in the correct sequence:

- Apply encryption (Encryption service) or apply a policy (Rights Management service) to a document before digitally signing 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.
- Certify a PDF document (Signature service) before you set usage rights (Reader Extensions service). If you certify a document after you apply usage rights, it could invalidate the usage rights signature, thereby removing the usage rights from the document.
- Digitally sign a PDF document (Signature service) after you set usage rights. Signing a PDF document after applying usage rights will 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.

The Execute Script service enables processes to execute scripts.

How developers use the Execute Script service

You can interact with the Execute Script service by developing a process in LiveCycle Workbench ES 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 gathered as a result of executing a script and transfer data to the LiveCycle ES server.
- Provide all context data for use during execution of a script.

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

The FTP service enables processes to interact with an FTP server. FTP service operations can retrieve files from the FTP server, put 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.

How developers use the FTP service

You can interact with the FTP service by developing a process in LiveCycle Workbench ES that uses the service. You can accomplish the following tasks using this 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 LiveCycle ES 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 LiveCycle ES 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 *LiveCycle Workbench ES Help*.

Development considerations

There are a few factors to consider when developing processes that use this service:

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

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

Files are commonly used to integrate with different systems. A process may need to output files in different formats such as XML, comma-delimited text, and PDF in order for other systems to process the data. A process can make use of the File Utilities 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.

How developers use the File Utilities service

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

- Save data as files on the file system.
- Retrieve information from files and save it as process data.
- Manipulate directories and files on the file system.

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

Development considerations

There are a few factors to consider when developing processes that use this service:

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

The Form Augmenter service enables a PDF form or Acrobat form to function within Adobe LiveCycle Workspace ES.

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

- Buttons will appear hidden when displayed in Workspace ES and 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 ES, which acts as an intermediary between the LiveCycle ES server and the form.
- Forms can be used both offline and online.

How developers use 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:

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

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

14

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 Adobe LiveCycle Designer ES) 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 may also contain form field data and an embedded PDF file. An XDP file generated by Designer ES can only be used if it carries an embedded base-64-encoded PDF document.

How developers use the Form Data Integration service

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

Fin@nce
corp.

MORTGAGE APPLICATION

Applicants: Complete this form for a mortgage application. One of our representatives will contact you within two business days.

Step 1: Mortgage Information

Property Sale Price: \$300,000.00	Down Payment \$5,000.00	Mortgage Amount: \$295,000.00
Term (Years): 25 <input type="text"/>	Closing Date: 01/26/2007	Monthly Mortgage Payment: \$1,724.54
Interest Rate: 5.00 <input type="text"/>		

Step 2: Applicant Information

Last Name Johnson	First Name Jerry	Middle Initial(s) D
Social Security Number 5 6 5 6 5 6 5 6 5 6	Phone Number (555) 555-0000	Date of Birth 26/8/1973
Mailing Address JJohnson@NoMailServer.com		
City New York	State New York <input type="text"/>	Zip Code 00501

In order 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 *LiveCycle Workbench ES Help*. For information about developing client applications that programmatically interact with this service, see *LiveCycle ES SDK Help*.

The Forms service enables you to create an interactive data capture client application that validates, processes, transforms, and delivers forms typically created in Adobe LiveCycle Designer ES. Form authors can develop a single form design that the Forms service can render in PDF, SWF, or HTML format in a variety of browser environments.

When an end user requests a form, a client application, such as a Java servlet, sends the request to the Forms service, which 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 and then delivers the form in a format that best matches the presentation and form filling capabilities of the target browser.

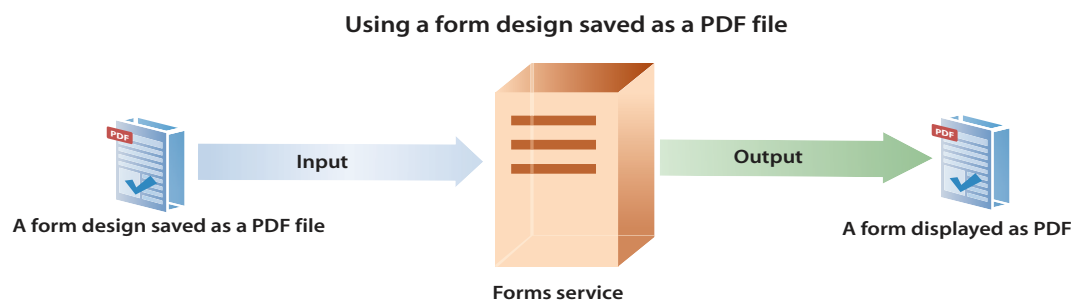
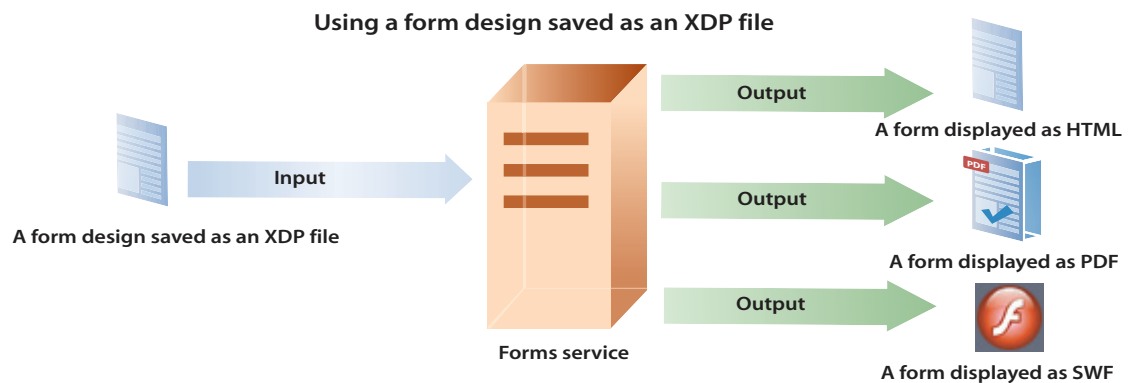
The Forms service performs the following functions:

- 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 resulting 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. For information about the properties and methods supported in each transformation, see the *Transformation Reference* at http://www.adobe.com/go/learn_lc_transformation.
- Dynamically generates PDF content, SWF content, or HTML content based on a form design with or without data. An HTML form can deliver multipage forms page by page. In contrast, a PDF form delivers all the pages at once. In Designer ES, the form author can script the current page number in the form design. The Forms service can merge one page of data submitted at a time or merge only the single page into the form design.
- Supports dynamic subforms created in Designer ES. The Forms service adds extra fields and boilerplate as a result of merging the form design with data or as a result of scripting. In the case of HTML, the added subforms can grow to unlimited page lengths. In the case of PDF, the added subforms paginate at the page lengths 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 desired form fragment. When a form author updates a fragment, all forms that contain a reference to the fragment will reflect the changes (when the form is rerendered). You can render a form that is based on a fragment like form designs that are not based on fragments. For information about creating fragments, see *LiveCycle Designer ES Help*.
- Validates data entry by performing calculations, accessing databases, or enforcing business rules on field-level data.
- Renders forms with file attachments. Likewise, 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 has been passed in by the application. 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 ES 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.

Development considerations

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



As shown in this diagram, if the form design is saved as an XDP file, the Forms service can output a form that is displayed as either PDF, form guides (SWF format), or HTML. However, if the form design is saved as a PDF file, the Forms service can only output a form that is displayed as PDF. That is, the Forms service cannot output an HTML form or form guides if the form design is saved as a PDF file.

The first step in planning your application is to determine the output format of the forms. If you want the Forms service to output PDF forms, form guides, or HTML forms, save your form designs as XDP files. If you want the Forms service to only output forms as PDF, save your form designs as PDF or XDP files. Next, plan the content of your form designs. Form design content varies from simplistic form designs that contain text and text box fields to complex form designs that contain multiple pages, different controls (such as radio buttons and drop-down lists), and scripts.

Creating form designs for the Forms service

Behavioral differences exist between form designs that are used to render PDF and HTML. Because form designs that are rendered as PDF are viewed using Adobe Acrobat or Adobe Reader, the form supports a full range of object properties that you define in the form design. If you want to render a form design as a form guide, then the Adobe Flash® player must be installed on the client computer.

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

1. Consult the *Transformation Reference* to determine how objects behave in a particular client device.
2. If you are designing a static form and want to output the form as HTML, you must enable transformation caching. For information, see the *LiveCycle Designer ES Help*.
3. When creating a form design, try to work around any 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 formats.
5. Read the section in *LiveCycle Designer ES Help* that discusses creating accessible forms 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 that you include in a form design should 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 only available 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 using Designer ES 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.

Processing requests

This section describes how the Forms service processes requests such as a form request, 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 series of specific processes and interactions with the client application that uses the Forms service Client API, the Forms service, and the client device, typically a web browser. The client application uses the Forms service Client API to render a form to a client device.

The following 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	Creates a <code>FormsServiceClient</code> and invokes the rendering method.	No action
No action	No action	Retrieves the form design, which is specified by the rendering method's <code>formQuery</code> parameter.
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.
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 by the rendering method's <code>formPreference</code> parameter.
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

User actions	Client application actions	Forms service actions
<p>The web browser performs these actions:</p> <ul style="list-style-type: none"> • 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. <p>Note: These actions only occur if the form is rendered as HTML.</p>	No action	No action
Views the form as either PDF, SWF, 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 is typically accomplished through the use of buttons that are located on the form design. 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 ES 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 a wrong value into a field, the user may have to correct the value before the form data can be submitted to the Forms service. 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 prior to a submit operation.

Submit button

A button can submit form 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 Client API, can retrieve the data.

A PDF form can submit four types of data (XDP, XML, PDF, and URL-encoded data). An HTML form only submits 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 ES. For information, see the *LiveCycle Designer ES Help*.

The content type of submitted PDF data is `application/pdf`. In contrast, the content type of submitted XML data is `text/xml`.

The following 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 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 LiveCycle Designer ES or by the <code>targetURL</code> data member that is defined in a <code>URLSpec</code> object passed to a render method such as <code>renderPDFForm</code>).	No action	No action
No action	Creates a <code>FormServiceClient</code> object and invokes the <code>processFormSubmission</code> method.	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 <code>validate</code> , <code>formatTest</code> , and <code>nullTest</code>).
No action	No action	Executes the form's <code>Close</code> event.
No action	No action	If this validation process fails, it indicates that at least one error exists. The returned processing state value is <code>Validate</code> .

User actions	Client application actions	Forms service actions
No action	Verifies that the Forms service returned a processing state value of <code>Validate</code> . In this situation, the result is sent back to the client browser so that the user can correct the mistake.	No action
<p>For forms that are displayed as HTML, the end user sees the form containing the same data, calculations, and list of errors to correct before resubmitting.</p> <p>For form guides (displayed as SWF), the end user sees the form containing the same data, calculations, and list of errors to correct before resubmitting.</p> <p>For forms that are displayed as PDF, a user interface is not defined. Validation errors can be retrieved by using the <code>FormsResult</code> object's <code>getValidationErrorsList</code> method.</p>	No action	No action
No action	No action	If the validation process succeeds, the processing state value is set to <code>Submit</code> .
No action	<p>Verifies that the Forms service returns a processing state value of <code>Submit</code>.</p> <p>Acknowledges that all form processing is complete.</p> <p>Any 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 process.</p>	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 that is located in the form design and renders the form back to the web browser with the results displayed in the form.

The following 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 calculation operation.

User actions	Client application actions	Forms service actions
<p>A user clicks a button that is located on a form.</p> <p>If the button's <code>Click</code> 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.</p> <p>A form guide implements an <code>XFA Subset</code> in <code>ActionScript</code>, which runs in the Adobe Flash player.</p> <p>If the button's <code>Click</code> event is marked run on server, the form is submitted to the Forms service.</p>	No action	No action
No action	Creates a <code>FormsServiceClient</code> object and invokes the <code>processFormSubmission</code> 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 sent back 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

The Generate PDF service converts files in a variety of native formats to PDF documents, and converts PDF documents to a number of file formats.

The Generate PDF service enables you to convert the following document types to PDF documents:

- HTML
- Microsoft Word
- Microsoft Powerpoint
- Microsoft Excel
- Microsoft Project
- Microsoft Visio
- AutoCAD
- Any third party generic application type for which you have a PDF generating application

In addition, the Generate PDF service can convert PDF documents to these file formats:

- Encapsulated PostScript (EPS)
- HTML
- DOC (Microsoft Word format)
- RTF
- Text (both accessible and plain)
- XML

How developers use the Generate PDF service

When converting to PDF and other file formats, the conversion operation allows you to specify several options to apply to the resulting document. For example, you can specify the following options:

- Preconfigured Adobe PDF settings, such as high quality print, oversized pages, and press quality.
- Preconfigured security settings for applying password-based encryption to the PDF document.
- Custom job configuration settings to apply while generating the PDF document, such as optimizing for fast web viewing. This optional parameter contains the file with settings to apply while generating the PDF document (for example, optimization for web view) and settings to apply when the PDF document is created (for example, initial view and security).
- Preconfigured custom file type settings.
- Metadata information to embed in the generated PDF document. Metadata contains information such as the author of the document, the subject, and any keywords to associate with the document. Only UTF-8 encoded Adobe Extensible Metadata Platform (XMP) metadata is supported. For details of the format and for the specifications, please visit http://www.adobe.com/go/learn_lc_XMP.

For information about developing processes that use this service, see *LiveCycle Workbench ES Help*. For information about developing client applications that programmatically interact with this service, see *LiveCycle ES SDK Help*.

How administrators use the Generate PDF service

You can use the LiveCycle Administration Console to manage user access and set run-time properties that control the way that the Generate PDF service operates.

From the LiveCycle Administration Console, you can configure this service on the LiveCycle ES PDF Generator page. You can specify default PDF settings and security settings to apply when converting to PDF.

For information about administrative tasks, see *Administering LiveCycle ES*.

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 there is already a stored procedure that populates the database, 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 could query the database and look up the status based on the customer ID, and create a rule based on this value.

How developers use the JDBC service

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

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

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

Development considerations

The data source used to connect to the data server must be defined on the application server that hosts the LiveCycle ES server. The default value is the JNDI name of the data source for the LiveCycle ES 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.

The JMS service enables interaction with Java Messaging System (JMS) providers that implement both point-to-point messaging and publish/subscribe messaging.

The JMS service can receive a message from a designated message queue. For example, a process may be initiated based on a message being placed in a queue when an order is created. The JMS service, in this example, is used as the first step in the process to listen for the message and initiate the process.

The JMS service can also send a message to a designated message queue or publish a message to a topic. For example, there may be a requirement to send a message in a queue that informs other systems to “Create new customer profile” and pass the customer information. A process can leverage the JMS service to send the message and pass the customer details as an XML document from a process variable.

How developers use the JMS service

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

- Publish a message to a topic on the JMS provider
- Retrieve a message that is stored in a queue on the JMS provider
- Send a message to a queue on the JMS provider

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

Development considerations

There are factors to consider when developing processes that use this service:

- Queues and topics must be configured on the JMS provider (by an administrator) before you can use them. Typically, JMS applications use Java Naming and Directory Interface (JNDI) services for finding the names of the topics and queues that are configured. After finding the name, the JMS client connects to the JMS service to interact with the queues and topics.
- You need to configure the JMS service with default properties so that the service operations can connect and interact with a JMS provider and an associated JNDI service. The values of the service properties are set to default values based on the JBoss® application server. You need to change these values if you are using a different application server to host LiveCycle ES.

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 could make use of 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.

How developers use the LDAP service

You can interact with the LDAP service by developing a process in LiveCycle Workbench ES that uses the service. You can accomplish the following tasks 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 *LiveCycle Workbench ES Help*.

Development considerations

There are factors to consider when developing processes that use this service:

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

The Output service enables you to output documents in PDF (including PDF/A documents), PostScript, Printer Control Language (PCL), and Zebra Printer Language (ZPL) formats. Using the Output service, you can merge XML form data with a form design created in Designer ES and output the document to a network printer, a disk file, or when used as part of a process, to an email recipient as a file attachment.

The Output service accepts synchronous or asynchronous document requests from multiple client applications and accepts form data as single records or batch records. The service also supports data being uploaded or URI accessible to the LiveCycle ES server.

How developers use the Output service

You can accomplish the following tasks using this service:

- Output documents as PDF and PDF/A documents.
- Output print streams to network printers or a Line Printer Daemon (LPD) URI when the network has an LP daemon running.
- Create multiple output files based on the records in the form data.
- Create search rules to enable dynamic use of different form designs that can be applied to form data or generate document packages.
- Convert (flatten) an interactive PDF document to a non-interactive PDF document.
- Convert a PDF document to a PDF/A document.
- Define request details within an XML print specification.
- Save print results that can be retrieved later by using a request identifier.

For information about developing processes that use this service, see *LiveCycle Workbench ES Help*. For information about developing client applications that programmatically interact with this service, see *LiveCycle ES SDK Help*.

Development considerations

When creating a client application that invokes the Output service, you must consider several factors. For example, decide whether you want to use the Output service to generate document output as files, send documents to printers, or email documents to email recipients as file attachments. Also, you must consider the format of the document.

Form data

The Output service accepts both a form design that is typically created in Designer ES 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, as long as all XML elements are specified.

Form type

It is recommended that you use an XDP file as input, although in some cases, a PDF file can be used. However, when using a PDF file as input, there are the following limitations:

- A PDF file cannot be converted to PostScript, PCL, and ZPL formats.
- If the input PDF file is based on an XFA form and it is signed or certified, then data will not be merged into the form.
- 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.

Typically, a PDF file is used as input when transforming an interactive PDF file to a non-interactive PDF file. Also a PDF file can be transformed to a PDF/A file; however, the PDF file must be based on an XFA form, not an Acrobat form.

Email support

For email functionality, you must create a process in Workbench ES that uses the Email service. A process represents a business process that you are automating. You cannot programmatically send email messages using the Output service API. For information about creating processes, see *Workbench ES Help*.

Print specification

A *print specification* is an XML file that describes the request that is sent to the Output service. By configuring a print specification, you instruct the Output service to perform document output tasks such as creating a PDF/A document or sending a print stream to a network printer. The print specification is deprecated in LiveCycle ES 8.0.

When you invoke the Output service, it is unnecessary to create a print specification because you can set required values using service operations. Even if you choose to use the print specification, you still have to set values using the service operations. As a result, it is recommended that you set all values using service operations and not use a print specification.

ZPL output

You can use Designer ES to create a form design for ZPL print streams. However, Designer ES does not have preconfigured ZPL page sizes. It is recommended that you create and thoroughly test a form design that is used for ZPL print streams.

The default font for a ZPL form is CG Triumvirate. If you want to use more than one font in a ZPL print stream, you must perform font mapping. For information about mapping fonts, see *Output Administration Help*.

Printable areas

The default 0.25-inch non-printable margin is not exact for label printers and varies printer by printer, label size by label size. It is recommended that you keep the 0.25-inch margin or reduce it. However, it is strongly recommended that you do not increase the nonprintable margin. If you do, information in the printable area may not print correctly.

Always ensure that you use the correct XDC file for the printer. An easy mistake is to choose an XDC file for a 300-dpi printer and send the print stream 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 *LiveCycle Designer ES Help*.

Device profile (XDC file)

A device profile (XDC file) is a printer description file in XML format that enables the Output service to output documents as PostScript, PCL, and ZPL formats. The following XDC files are for use with the Output service and represent generic print description language support:

- `hppcl5c.xdc` - For use with PCL printers that support the HP PCL 5c printer language (monochrome)
- `hppcl5e.xdc` - For use with PCL printers that support the HP PCL 5e printer language (color)
- `ps_plain.xdc` - For use with printers that support the PostScript printer language (monochrome and color)
- `zpl203.xdc` - For use with 203 dpi (8 dots/mm) Zebra label printers (monochrome)
- `zpl300.xdc` - For use with 300 dpi (12 dots/mm) Zebra label printers (monochrome)

It is not necessary to modify these files for the Output service to create print streams. However, you can modify them to meet your business requirements. For information about modifying XDC files, see the *XDC Editor Help*. After you modify an XDC file, you reference it as part of developing a client application. For information about referencing an XDC file, see *LiveCycle ES SDK Help*.

In addition, the following 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 using device profiles. The samples are also a starting point for similar printers in the same product line.

- `hp4350pcl5e.xdc` - HP 4350 printer device profile using PCL5e (monochrome)
- `hp4350ps.xdc` - HP 4350 printer device profile using Postscript (monochrome and color)
- `lmt644pcl5e.xdc` - Lexmark T644 printer device profile using PCL5e (monochrome)
- `lmt644ps.xdc` - Lexmark T644 printer device profile using Postscript (monochrome and color)

The PDF Utilities service can convert between PDF and XDP file formats, set and retrieve PDF document properties, and determine the save mode of PDF documents.

The PDF Utilities service is useful in processes to determine certain properties about a PDF document, perform document conversions, and manipulate XMP metadata. For example, before converting a PDF document to another format, it is useful to inspect its properties in order to determine which service operation to invoke for the conversion.

How developers use the PDF Utilities service

You can accomplish the following tasks using this service:

- Convert PDF documents to XDP files and vice versa.
- Determine properties about PDF documents such as the minimum Acrobat version required to read them.
- Set and retrieve the save mode (such as incremental save or fast web viewing) of PDF documents. This means that you can ensure that a PDF document is saved incrementally to reduce the time required to save, for fast web viewing, or using a full save (without optimizations).

For information about developing processes that use this service, see *LiveCycle Workbench ES Help*. For information about developing client applications that programmatically interact with this service, see *LiveCycle ES SDK Help*.

The Reader Extensions service enables your organization to easily share interactive PDF documents by extending the functionality of Adobe Reader. The Reader Extensions service fully supports any PDF document, up to and including PDF 1.7. It works with Adobe Reader 7.0 and later. The service adds usage rights to a PDF document, activating features that are not usually available when a PDF document is opened using Adobe Reader. Third party users do not require additional software or plug-ins to work with the rights-enabled documents.

The following table lists the usage rights and indicates the actions that users can perform in Adobe Reader when each right is enabled.

Name of usage right	User can perform this action
enabledBarcodeDecoding	Use barcoded forms within Adobe Reader.
enableComments	Add comments within Adobe Reader.
enableCommentsOnline	Add comments within Adobe Reader while online.
enableDigitalSignatures	Sign a PDF document within Adobe Reader.
enableDynamicFormFields	Add and work with dynamic form fields within Adobe Reader.
enableDynamicFormPages	Add and remove pages to a PDF document within Adobe Reader.
enableEmbeddedFiles	Attach files to a PDF document within Adobe Reader.
enableFormDataImportExport	Import and export form data from a PDF document from within Adobe Reader.
enableFormFillIn	Fill in form fields and save the PDF document from within Adobe Reader.
enableOnlineForms	Interact with a form while online from within Adobe Reader.
enableSubmitStandalone	Submit information from a form within Adobe Reader.

How developers use the Reader Extensions service

You can accomplish the following tasks using this service:

- Apply usage rights to PDF documents.
- Get usage rights information from a PDF document or credential.
- Remove usage rights from PDF documents.

For information about developing processes that use this service, see *LiveCycle Workbench ES Help*. For information about developing client applications that programmatically interact with this service, see *LiveCycle ES SDK Help*.

Development considerations

There are a number of factors to consider when developing an application that uses the Reader Extensions service.

Configuring the credential

In order to apply usage rights to PDF documents, you must have a valid credential. A credential was configured during the installation of LiveCycle ES. For information about configuring a credential, see *Administering LiveCycle ES* or consult with your application server administrator.

Order of operations

If you are working with multiple services, it is important to perform service operations in the correct sequence:

- Apply encryption (Encryption service) or apply a policy (Rights Management service) to a document before you digitally signing 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.
- Certify a PDF document (Signature service) before you set usage rights (Reader Extensions service). If you certify a document after you apply usage rights, it could invalidate the usage rights signature, thereby removing the usage rights from the document.
- Digitally sign a PDF document (Signature service) after you set usage rights. Signing a PDF document after applying usage rights will 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.

Adding usage rights to interactive forms

Sometimes it is necessary to add usage rights to a PDF document while working with interactive forms created in Designer ES. For example, assume that you create an interactive form using Designer ES and you want to use a form script to reference a data connection SOAP endpoint where form data is retrieved. Next assume that you want Adobe Reader 8.0 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 ES) being used. In order to set the data connection SOAP endpoint using a form script, you must add the `enableDynamicFormFields` usage right to the interactive form.

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, then the PDF document is no longer compatible with that version of Acrobat (or Adobe Reader). The PDF document is only compatible with Adobe Reader 7.0 or greater.

The Repository service provides storage capabilities. When a developer creates an application, the developer can deploy the assets in the repository instead of on a file system. The assets may consist of XML forms, PDF forms (including Acrobat forms), form fragments, images, processes, profiles, policies, SWF files, DDX files, XML schemas, WSDL files, and test data.

The repository tracks the version of each asset in a LiveCycle ES application. At runtime, services can retrieve assets from the repository as part of completing an automated business process.

How developers use the Repository service

You can use this service in a process to retrieve resources from the repository. The Repository 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 using the Repository API.

For information about developing a process that uses this service, see *LiveCycle Workbench ES Help*. For information about developing a client application that programmatically interacts with this service, see *LiveCycle ES SDK Help*.

24 | Rights Management Service

The Rights Management service enables users to dynamically apply confidentiality settings to PDF documents and to maintain control over the documents, no matter how widely they are distributed. The Rights Management service also protects other file types such as Microsoft Word files (DOC files), Microsoft Excel files, and CATIA files. However, you cannot use the Rights Management Client API to work with these file types.

The Rights Management service prevents information from spreading beyond the user's reach by enabling the users to maintain control over how recipients use the policy-protected PDF document. A user can specify who can open a document, limit how they can use it, and monitor the document after it is distributed. A user can also dynamically control access to a policy-protected document and can even dynamically revoke access to the document.

Using client applications (such as Acrobat Professional and Acrobat Standard), users can protect PDF documents by applying security policies (a collection of predefined user access and confidentiality settings).

When working with PDF files, users can use Acrobat or Adobe Reader to access policy-protected documents. Client applications also communicate with the server component to authenticate end users, provide event information to the server for auditing purposes, and determine if access to documents is authorized.

About policies and policy sets

A policy defines a set of security permissions and users who can access a PDF document to which the policy is applied. A policy also enables the permissions on a document to be changed dynamically. It enables the person who protects the document to change the security settings, to revoke access to the document, or to switch the policy.

Policy sets are used to group a set of policies that have a common business purpose. These policy sets are then made available to a subset of users in the system.

Each policy set has an associated policy set coordinator. The policy set coordinator is an administrator or user with additional permissions. The policy set coordinator is typically a specialist in the organization who can best author the policies in a given policy set.

Policy sets are created and deleted in the Rights Management ES administrator web pages by super users and policy set administrators who have been granted permission to do so.

When a policy set is deleted, all policies in the policy set are deleted as well. All documents protected using a policy in the deleted policy set are revoked.

On installation of Rights Management ES, a default policy set is created called Global Policy Set. This policy set is managed by the administrator who installed the software.

Security methods and technology

This section describes the security methods and technology that Rights Management ES implements. To ensure the confidentiality of documents that are protected by policies, Rights Management ES implements three layers of security:

- Authentication
- Authorization
- Document confidentiality

Authentication

All users are required to log in to interact with Rights Management ES. Users must log in before performing the following tasks:

- Opening the Rights Management ES web application in a web browser
- Securing documents with policies in a supported client application
- Opening policy-protected documents

Methods of authentication

Rights Management ES supports two methods of authentication:

- Username/Password. Users are prompted for their user name and password.
- Kerberos (from Acrobat on Microsoft® Windows® only). Enables Acrobat or Adobe Reader users on a Windows platform to be transparently authenticated.

Note: Rights Management ES supports authentication over SSL connections.

Users can be internal or external to your organization. Internal users have corresponding user records in your organizational user directory. Rights Management ES authenticates internal users against that user directory.

External users can register with Rights Management ES and create user accounts. Rights Management ES stores the external user accounts in the database and uses the accounts to authenticate external users.

User accounts for Rights Management ES administrators are also stored in the database.

SAML authentication assertions

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

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

Subsequent messages sent to Rights Management ES 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, Rights Management ES does not support third-party SAML assertions.

Logging in through Acrobat

When Rights Management ES authenticates an Acrobat user, the server returns the SAML authentication assertion to Acrobat.

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

Authorization

Rights Management ES 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. Rights Management ES implements the roles Administrator, User, and External User.

Administrator

Users are assigned the role of Administrator when an administrator account is created for them. Only administrators can create other administrator accounts. Administrator accounts are not based on LDAP records but are stored in the relational database.

Through the web application, administrators have complete access to the server configuration and can manage all aspects of policies, policy-protected documents, external users, administrator accounts, and event audits.

Users cannot use their administrator account to log in through Acrobat. Administrator accounts cannot be used to protect documents with policies or open policy-protected documents.

User

Users are assigned the role of User if they have a corresponding record in a user directory that Rights Management ES uses for authentication.

Through the web application, users can create and manage their own policies, the policy-protected documents that they have distributed, and the events that are associated with those documents.

Users can also log in through Acrobat to use the Rights Management ES client features that Acrobat provides, such as securing documents with organizational policies or with the users' own policies that they created.

External User

Users are assigned the role of External User when they manually register with Rights Management ES. External users can create a Rights Management ES user account when an administrator explicitly invites them or when they are added to a policy.

Rights Management ES configuration options determine the features that external users can access. Administrators can provide external users with access to all the features that internal users can access or provide them access to a limited set of features. Administrators can also permanently or temporarily remove any external user.

External users can also log in through Acrobat to use the Rights Management ES client features that Acrobat provides.

For more information about setting up the registration process for external users and configuring access options, see the *LiveCycle Rights Management ES Help*.

Document confidentiality

Rights Management ES uses several technologies to protect documents and to provide access to them. In general, Rights Management ES uses a symmetric cryptographic key system for encryption. Client applications such as Acrobat perform document encryption. Documents are never sent to Rights Management ES. Rights Management ES encrypts policies and licenses that are associated with documents.

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.

Policies contain information about the authorized users and the confidentiality settings to apply to PDF documents. Users can be any user in your organization, as well as people who are external to your organization who have registered with Rights Management ES or for whom the administrator has created an account. If the administrator enables the user invitation feature, it is even possible to add new invited users to policies, thereby initiating a registration invitation email process.

The confidentiality settings you specify in a policy determine how the recipients can use the PDF 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.

When you apply a policy to a PDF document, the information that the document contains, including any files (text, audio, or video) that you save in the document, is protected by the confidentiality settings specified in the policy.

Note: Document confidentiality settings that are applied through a policy override any settings that may have been applied to the PDF document in Acrobat by using the password or certificate security options. See the Acrobat Help for more information.

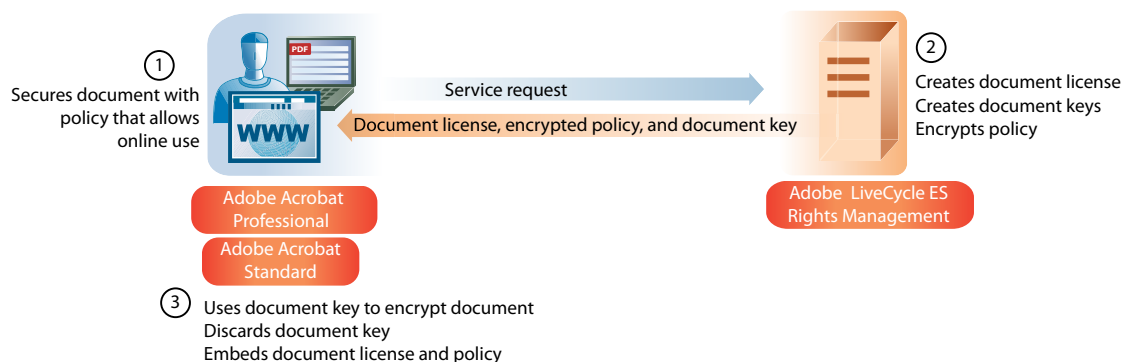
Document protection for online use

Policies can be designed so that users must have a network connection to open the documents they protect. Securing documents for online use employs a straightforward process for encrypting the document and providing access only to authenticated and authorized users.

When users are online, Rights Management ES records events in the audit log as users interact with Rights Management ES or policy-protected documents. For information about the audit log, see the *LiveCycle Rights Management ES Help*.

Users and administrators create policies through the Rights Management ES web pages, and users can also create them using Acrobat. Only one policy at a time can be applied to a PDF document. You can apply a policy using one of the following methods:

- Open the document in Acrobat 7.0, use the appropriate menu to log into Rights Management ES, and select or create a policy.
- Open the document in Acrobat 8.0 and you will then be redirected to a web page where you can create or edit a policy.
- Send a document as an email attachment in Microsoft Outlook. In this case, you can select a policy from a list of existing policies or select an auto-generated policy that Acrobat creates with a default set of permissions to protect the document only for the recipients of the email message.



The following table describes the steps in this diagram.

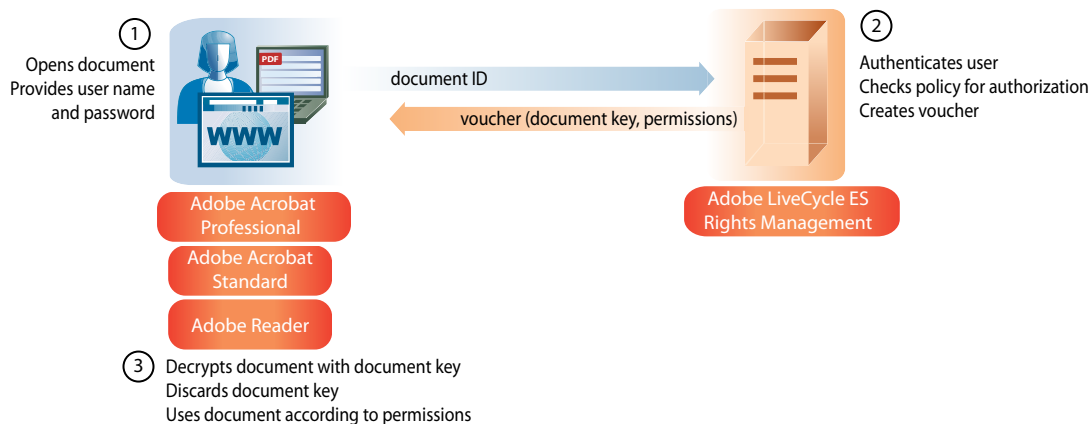
Step	Description
1	The document owner or administrator protects the document with a policy that allows online use. Users can apply policies to documents using any supported client application such as Acrobat 7.0 or later and Microsoft Word or Excel. Developers can also protect documents with policies by using the Rights Management service in a process or programmatically by using the Rights Management service API.
2	Rights Management ES creates a document license and document key, and encrypts the policy. The document license, encrypted policy, and document key are sent back to the client application. The document key is a symmetric key for encrypting the document. Each protected document has an associated document key. The document license is an XML document that identifies the protected PDF document, the policy, and the identity of the server. The server digitally signs the license to ensure data integrity.
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.

Document access for online use

To open and use policy-protected documents, the policy must include the document user's name. The document user also needs Acrobat or Adobe Reader, and a valid Rights Management ES account.

To open documents that are protected for online use, a user must first log into Rights Management ES. After Rights Management ES determines the document permissions to grant the user, it sends the document key to the client for decrypting the document. The key is packaged in a voucher along with the document permissions.

If the policy specifies that document events should be logged, as the user opens and uses the document, the client software immediately sends event information to the server for logging. For information about the audit log, see the *LiveCycle Rights Management ES Help*.



The following table describes the steps in this diagram.

Step	Description
1	The document user opens the document and provides a user name and password. This task is performed in the supported client application. The document identifier is sent to the Rights Management service.
2	The Rights Management service authenticates the user, checks the policy for authorization, and creates a voucher. The voucher (containing the document key and permissions) is sent back 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 permissions of the policy. These tasks are performed in the supported client application.

If the user saves a copy of a policy-protected document by using the Save or Save As command, 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 for 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 ability to open the document or changes the policy

Document protection for offline use

Policies can be designed so that users can open documents when they do not have a network connection to Rights Management ES.

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

In general, to open documents that are protected with offline policies, users can open the documents offline if they have recently logged into Rights Management ES. To enable opening documents offline, when users log in, their client software synchronizes with Rights Management ES.

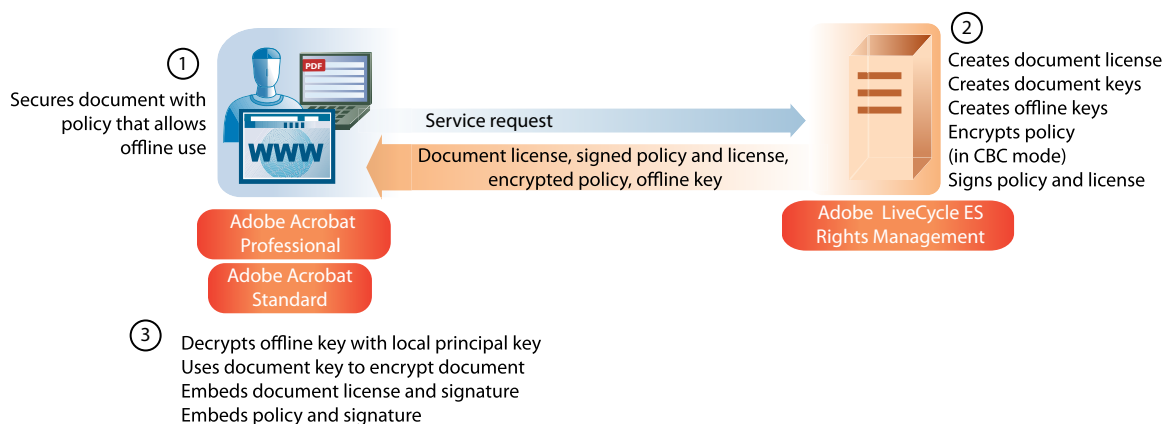
Although securing documents for offline use employs a more rigorous process for ensuring document security, the experience of a user is generally the same as for online use.

Protection for offline use

Users protect documents for offline use by using the same procedure as for securing documents for online use. Rights Management ES implements additional measures to ensure that keys are not compromised when they are sent to clients. In addition to the document license and the policy, Rights Management ES returns the following items to the client when users protect a document for offline use.

Item	Description
Offline key	The document key encrypted with the principal key. Principal keys are symmetric keys that Rights Management ES creates for each user included in offline policies. Each user is associated with a unique principal key. Note: Key rollovers of principal keys can be performed as required and on a scheduled basis.
Digital signatures	The encrypted message digests of the document license and the policy. Both are encrypted with the offline key.

The client software discards the document key and offline key after encrypting the document.



Synchronization for offline use

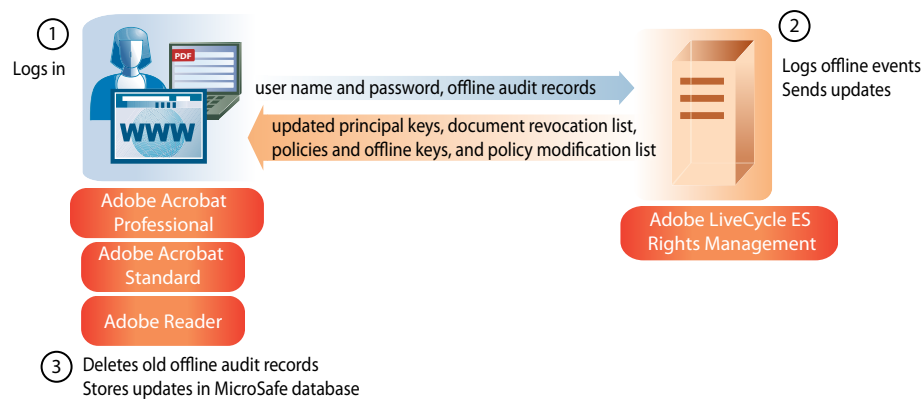
Synchronization enables users to open policy-protected documents when they do not have a connection to Rights Management ES. Synchronization updates and securely stores the following information about the client computer:

- Principal keys and offline keys—Synchronization of keys is required if the principal keys have been rolled over since the last time the user logged in.
- Document licenses and policies—Synchronization of licenses and policies is required because changes to policies can affect users' access to documents, and document access can be revoked or unrevoked.
- Identifications of documents that are no longer available for opening.

Note: Only the information that affects the current user is sent to the client.

If the offline policy specifies that document events should be logged, to maintain a complete event audit, the client software sends information about any events that occurred offline. For example, if a user has opened a document offline, when synchronization occurs, the client sends the event information to the server. On the Rights Management ES web pages, the event log indicates which events occurred offline.

Client software automatically synchronizes with Rights Management ES when users log in. Users can open documents offline if they have recently logged into Rights Management ES. The policy determines the length of time that can pass after synchronization that a user can open a document offline.



Document access for offline use

Policies determine if users can open policy-protected documents when the users are offline. To open a document offline, a user must have recently logged into Rights Management ES and must have closed the network connection. When a user attempts to open a policy-protected document offline, the client software performs the following operations:

- Reads the document license to determine if the user is permitted to open the document offline
- Decrypts the policy by using the offline key and determines the user permissions
- Validates the authenticity of the document license and policy by using the offline key and the signatures
- Decrypts the offline key by using the local copy of the principal key and obtains the document key
- Decrypts the document by using the document key and then opens the document
- Discards the document key

In addition, to open policy-protected documents offline, users must be authenticated by the operating system of their client computer.

Security standards and technology

The following table provides details about the methods that Rights Management ES uses to implement security.

Action	Technology or method used
Creating document keys Creating principal keys	Pseudo Random Number Generator (PRNG) generated in accordance with ANSI X9.61.
Creating Initialization Vectors (IV) for AES-128 encryption in CBC mode	The implementation used is the RSA BSafe Crypto-C (in Acrobat) or Crypto-J (in Rights Management ES) toolkits.
Creating offline keys	The 128-bit document key is encrypted with the 256-bit principal key by using Advanced Encryption Standard (AES) -256 in accordance with FIPS Publication 197. The offline key has a length of 128 bits.
Encrypting PDF documents	AES-128 in accordance with Federal Information Processing Standards (FIPS) Publication 197. Uses the 128-bit document key.
Encrypting policies	AES-128 (FIPS Pub 197) in Cipher Block Chaining (CBC) mode (FIPS Pub 81). Uses the 128-bit offline key.
Creating message digests	Secure Hash Algorithm-1 (SHA-1) in accordance with FIPS Pub 180-2. Produces 160-bit message digests.
Creating message authentication codes (MACs) for signing document licenses and policies	HMAC mechanism in conjunction with the SHA-1 hashing function. Rights Management ES uses the 256-bit principal key to generate the MAC. Frequency of key refreshment is configurable.
Encrypting with principal keys	AES-256 in accordance with FIPS Publication 197.
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.

How developers use the Rights Management service

When developing the process or the client application, you can specify any of these operations:

- Create new policies. You can create and save any number of policies, using security settings appropriate for different situations and users.
- Apply a policy to a PDF document in order to protect the document. By applying a policy to a PDF document, you restrict access to the document. While the document is open, you can also restrict access to Acrobat and Adobe Reader features, including the ability to print and copy text, make changes, and add signatures and comments to a document.
- Add watermarks to policy-protected documents. Watermarks help ensure the security of a document by uniquely identifying the document and controlling copyright infringement.
- Delete an existing policy. After a policy is deleted, it can no longer be used to protect documents. However, existing policy-protected documents that are using the policy are still protected.
- Modify an existing policy's attributes such as watermark, principals, and offline lease period.
- Revoke access to a policy-protected PDF document resulting in all copies of the document being inaccessible to users. When a user attempts to open a revoked PDF document, they are redirected to a specified URL where a revised document can be viewed.
- Reinstate access to a revoked PDF document, resulting in all copies of the revoked document being accessible to users. When a user opens a reinstated document that was revoked, the user is able to view the document.
- Remove a policy from a policy-protected document in order to remove security from the document.
- Track specific actions as they occur, such as applying a policy to a document, opening a policy-protected document, and revoking access to documents. You can search for specific events. By searching for events, you can perform tasks, such as creating a log file of certain events.

For information about using the Rights Management service in a process, see the *LiveCycle Workbench ES Help*. For information about using the Rights Management API, see the *LiveCycle ES SDK Help*.

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.

How developers use the Set Value service

You can interact with the Set Value service by developing a process in LiveCycle Workbench ES 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 the XMP Path Language (XPath) specification at <http://www.w3.org/TR/xpath>.

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

The Signature service enables an organization to protect the security and privacy of PDF documents that it distributes and receives. This service uses digital signatures and certification to ensure that documents cannot be altered by anyone other than the intended recipients. Because security features are applied to the document itself, the document remains secure and controlled for its entire life cycle; beyond the firewall, when it is downloaded offline, and when it is submitted back to the organization.

About public key technology

The Signature service uses public and private keys to sign and certify PDF documents. A private key is unique to a user and it is important that no one other than the user has access to this key. The other key is the user's public key, which can be freely shared with other users.

When you sign a document, a digest of the document contents is created using a hashing algorithm. A private key is used to encrypt the digest, which becomes part of the digital signature. Data encrypted with a private key can be successfully decrypted with the corresponding public key. That is, to validate the signature, a recipient must have the corresponding public key, which is used to decrypt the digest. The digest is then recalculated and compared with the decrypted document digest to make sure that the document has not been altered.

About certificates and credentials

A public-key certificate contains a user's public key, along with identifying information. The X.509 format is used for storing certificates. Certificates are typically issued and digitally signed by a certificate authority (CA), which 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; CRLs are published periodically by certificate authorities. Revocation status of a certificate can also be retrieved through Online Certificate Status Protocol (OCSP) over the network.

A credential contains a user's private key, as well as other identifying information, such as an alias and a password required to access (log into) the physical credential. The credential may exist in one of several forms:

- As files on disk in PKCS#12 format
- In a hardware security module (HSM); HSM credentials are retrieved using the PKCS#11 standard

Integrating with an existing security infrastructure

Digital Signatures ES ensures persistent privacy and security of your PDF documents through the application and processing of different types of digital signatures. Digital signing functions are performed by using public-key technology tools.

Digital Signatures ES can integrate your existing public key infrastructure. Trust Store Management is used to store and manage your certificates, credentials and certificate revocation lists. For information, see *Trust Store Management Help*. Additionally, you can use an independent HSM device to manage the storage of these objects.

The following table provides a summary of the technologies and industry standards that are supported by Digital Signatures ES and used for its various security functions:

Type of security	Control mechanism	Supported technology or standard
Integrity	One-way hash	SHA-1, SHA-256, SHA-384, and SHA-512 MD5 RIPEMD160
Authenticity	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) Timestamping (TSP) to validate the signing time
	Certificate validity	Certificate Revocation List (CRL) Online Certificate Status Protocol (OCSP) RFC 3280 compliant path validation

How developers use the Signature service

Using the Signature service, you can perform tasks such as:

- Add digital signature fields to a PDF document.
- Retrieve the names of signature fields located in a PDF document.
- Digitally sign PDF documents.
- Certify PDF documents.
- Validate digital signatures located within a PDF document.
- Remove a digital signature from a signature field.

For information about developing processes that use this service, see *LiveCycle Workbench ES Help*. For information about developing client applications that programmatically interact with this service, see *LiveCycle ES SDK Help*.

Development considerations

There are a number of factors to consider when using the Signature service operations.

Order of operations

If you are working with multiple services, it is important to perform service operations in the correct sequence:

- Apply encryption (Encryption service) or apply a policy (Rights Management service) to a document before you digitally signing 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.
- Certify a PDF document (Signature service) before you set usage rights (Reader Extensions service). If you certify a document after you apply usage rights, it could invalidate the usage rights signature, thereby removing the usage rights from the document.
- Digitally sign a PDF document (Signature service) after you set usage rights. Signing a PDF document after applying usage rights will 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.

Operation compatibility

Not all Signature service operations are compatible with all types of PDF documents. If you attempt to perform an operation on a PDF document that is not compatible with the operation, an exception is thrown. The following table lists PDF document types and Signature service operations.

Operation	PDF document (non XFA)	Dynamic XFA			
		Static XFA PDF document (non shell)	Dynamic XFA PDF document (non shell)	Static XFA PDF document (shell)	Dynamic XFA PDF document (shell)
Certify documents (visible signature field)	Supported	Supported (2)	Supported (2,3)	Not Supported	Not Supported
Certify documents (invisible signature field)	Supported	Supported (2)	Supported (2,3)	Not Supported	Supported (1,2,3)
Digitally sign documents	Supported	Supported (2)	Supported (2)	Not Supported	Not Supported
Verify signatures	Supported	Supported	Supported	N/A	Supported
Add signature fields	Supported	Supported	Not Supported	Not Supported	Not Supported
Add invisible signature fields	Supported	Supported	Not Supported	Not Supported	Not Supported
Clear Signature Field	Supported	Supported	Supported	N/A	Supported

Operation	PDF document (non XFA)	Static XFA PDF document (non shell)	Dynamic XFA PDF document (non shell)	Static XFA PDF document (shell)	Dynamic XFA PDF document (shell)
Remove Signature Field	Supported	Supported	Not Supported	Not Supported	Not Supported
Retrieve signature field names	Supported	Supported	Supported	Supported	Supported

The following information is applicable to the previous table:

1. No document snapshot is saved in such a signature, and only the document template is certified.
2. Field MDP is specified only when the XFA template is not applied while certifying or signing.
3. The signature field being certified is not locked even if the `Lock` parameter is set to `true`.

Note: For information about different PDF document types, see *LiveCycle Designer ES Help*.

The Stall service is useful for preventing situational errors that you anticipate may occur. This service provides the capability to stall the branch to which it belongs.

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.

How developers use the Stall service

You can interact with the Stall service by developing a process in LiveCycle Workbench ES 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 if the situational error has occurred. When the situation occurs, the branch is stalled so that you can fix the error and restart the process using the LiveCycle Administration Console.

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

The User service provides the capability to involve people in processes.

How developers use the User service

You can use this service in a process to create a task and assign it to a user.

You can configure the behavior of the task at run time, the features that people can use with the task when they open it in LiveCycle Workspace ES, or when they use email to complete the task. For example, you can use the service to perform the following tasks:

- Identify the input form data that is used for the task.
- Provide instructions that describe what the user must do to complete the task.
- Provide routes as options for the user to select when completing the task.
- Restrict to whom a user can forward the task and with whom to consult on the task.
- Specify whether a form must be saved to complete the task and the location to store output form data.
- Specify the manner in which attachments and notes are handled for the task, and where they are stored after execution.
- Configure reminders to send to users and deadlines for completing the task.

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

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 LiveCycle ES server. When a process stalls and is not functioning as expected, this 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.

How developers use the Variable Logger service

You can interact with the Variable Logger service by developing a process in LiveCycle Workbench ES that uses the service. When using this service in a process, you can specify:

- 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 filename and path of the log file or an XPath expression to a process variable that contains the filename and path in the case of 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 *LiveCycle Workbench ES Help*.

The Wait Point service provides the capability to delay the progression of a process at a step in a process.

How developers use the Wait Point service

You can interact with the Wait Point service by developing a process in LiveCycle Workbench ES that uses the service.

When you use this service in a process to delay the execution of the next operation in the process, you specify the amount of time to wait by providing values for days, hours, minutes, and seconds.

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

The Web 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 invokes a specified web service, passing through values for each of its parameters, and saves the return values from the operation into a designated variable within a process.

The Web service interacts with web services by sending and receiving SOAP messages. The service also supports sending MIME and MTOM attachments with SOAP messages using the WS-Attachment protocol.

How developers use the Web Service

You can interact with the Web Service by developing a process in LiveCycle Workbench ES that uses the service. You can accomplish the following tasks 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 then 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.
- 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 *LiveCycle Workbench ES Help*.

32 | 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.

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.

For more information about XMP, see <http://www.adobe.com/products/xmp/>.

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. In addition, users can add custom properties with a unique name and value.

- In the Metadata dictionary of the document catalog. This dictionary contains metadata 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 containing 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 the “Document Information Dictionary” section of 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, any values you specify will be overridden.

How developers use the XMP Utilities service

You can accomplish the following tasks using this service:

- Import metadata into a PDF document from an external XML file or from another PDF document.
- Export metadata from a PDF document then save it as a file.
- Modify the XMP in a number of ways. For example, you can add and remove metadata properties and set their values. When importing or exporting metadata, you can define these metadata values: author, creator, subject, keywords, title, and producer.

For information about developing processes that use this service, see *LiveCycle Workbench ES Help*. For information about developing client applications that programmatically interact with this service, see *LiveCycle ES SDK Help*.

The XSLT Transformation service enables processes to apply Extensible Stylesheet Language Transformations (XSLT) on XML documents.

How developers use the XSLT service

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

- Configure the XSLT Transformation 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 using an XSLT script that is stored as process data. The XML document is also stored as process data.
- Transform an XML document 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 see the result.
- Save the resulting XML document as process data.

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