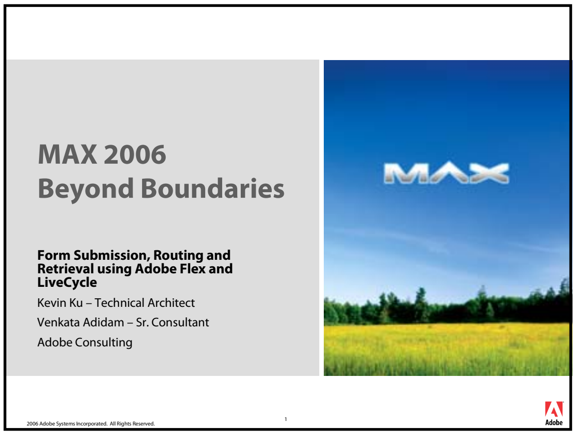
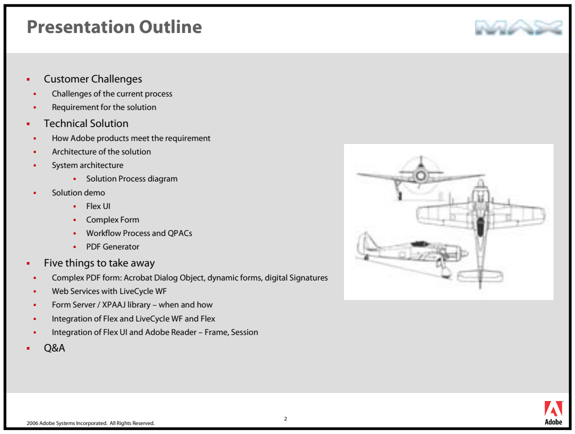


MAX 2006 Beyond Boundaries

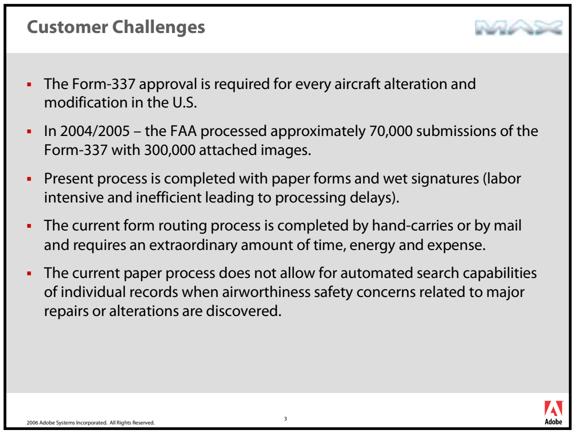
Form Submission, Routing and Retrieval using Adobe Flex and LiveCycle

Kevin Ku – Technical Architect
Venkata Adidam – Sr. Consultant
Adobe Consulting



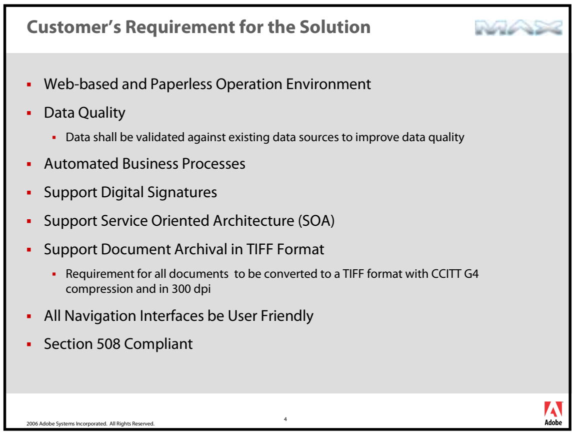
Presentation Outline

- Customer Challenges
  - Challenges of the current process
  - Requirement for the solution
- Technical Solution
  - How Adobe products meet the requirement
  - Architecture of the solution
  - System architecture
    - Solution Process diagram
  - Solution demo
    - Flex UI
    - Complex Form
    - Workflow Process and QPACs
    - PDF Generator
- Five things to take away
  - Complex PDF form: Acrobat Dialog Object, dynamic forms, digital Signatures
  - Web Services with LiveCycle WF
  - Form Server / XPAAJ library – when and how
  - Integration of Flex and LiveCycle WF and Flex
  - Integration of Flex UI and Adobe Reader – Frame, Session
- Q&A



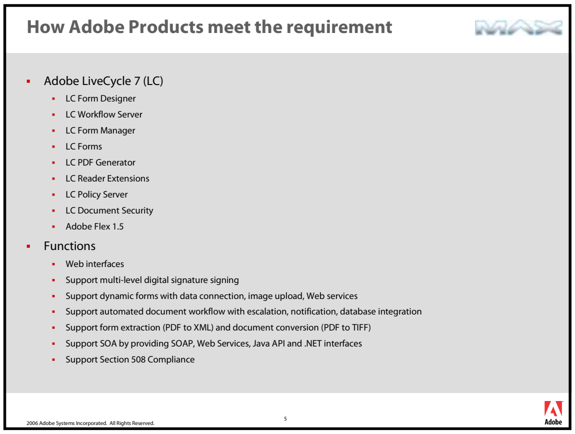
Customer Challenges

- The Form-337 approval is required for every aircraft alteration and modification in the U.S.
- In 2004/2005 – the FAA processed approximately 70,000 submissions of the Form-337 with 300,000 attached images.
- Present process is completed with paper forms and wet signatures (labor intensive and inefficient leading to processing delays).
- The current form routing process is completed by hand-carries or by mail and requires an extraordinary amount of time, energy and expense.
- The current paper process does not allow for automated search capabilities of individual records when airworthiness safety concerns related to major repairs or alterations are discovered.



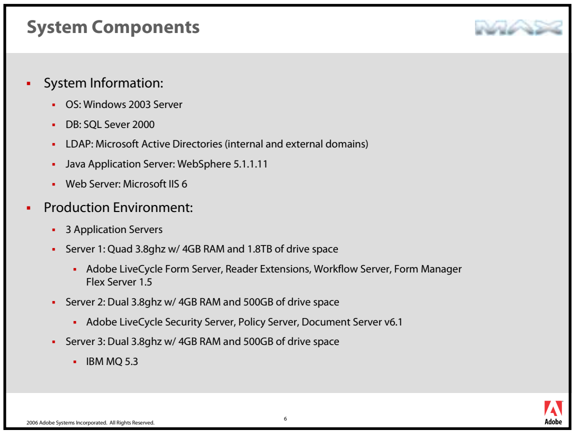
Customer's Requirement for the Solution

- Web-based and Paperless Operation Environment
- Data Quality
  - Data shall be validated against existing data sources to improve data quality
- Automated Business Processes
- Support Digital Signatures
- Support Service Oriented Architecture (SOA)
- Support Document Archival in TIFF Format
  - Requirement for all documents to be converted to a TIFF format with CCITT G4 compression and in 300 dpi
- All Navigation Interfaces be User Friendly
- Section 508 Compliant



How Adobe Products meet the requirement

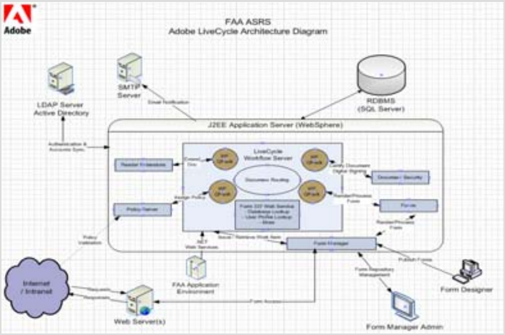
- Adobe LiveCycle 7 (LC)
  - LC Form Designer
  - LC Workflow Server
  - LC Form Manager
  - LC Forms
  - LC PDF Generator
  - LC Reader Extensions
  - LC Policy Server
  - LC Document Security
  - Adobe Flex 1.5
- Functions
  - Web interfaces
  - Support multi-level digital signature signing
  - Support dynamic forms with data connection, image upload, Web services
  - Support automated document workflow with escalation, notification, database integration
  - Support form extraction (PDF to XML) and document conversion (PDF to TIFF)
  - Support SOA by providing SOAP, Web Services, Java API and .NET interfaces
  - Support Section 508 Compliance



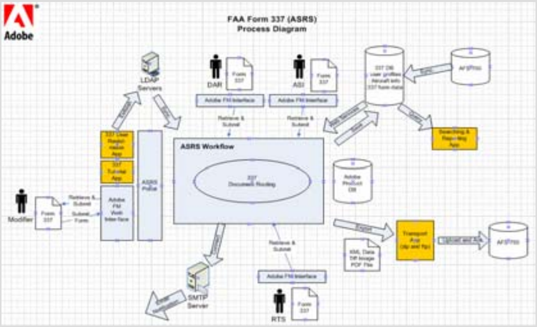
System Components

- System Information:
  - OS: Windows 2003 Server
  - DB: SQL Sever 2000
  - LDAP: Microsoft Active Directories (internal and external domains)
  - Java Application Server: WebSphere 5.1.1.11
  - Web Server: Microsoft IIS 6
- Production Environment:
  - 3 Application Servers
  - Server 1: Quad 3.8ghz w/ 4GB RAM and 1.8TB of drive space
    - Adobe LiveCycle Form Server, Reader Extensions, Workflow Server, Form Manager Flex Server 1.5
  - Server 2: Dual 3.8ghz w/ 4GB RAM and 500GB of drive space
    - Adobe LiveCycle Security Server, Policy Server, Document Server v6.1
  - Server 3: Dual 3.8ghz w/ 4GB RAM and 500GB of drive space
    - IBM MQ 5.3

System Architecture



System Process Diagram



Solution Demo

- User Interfaces
  - Flex interface
  - Adobe LiveCycle Form Manager interface
- Complex Form
  - Demo dynamic forms function
  - Web service calls
  - Dialog box in PDF
  - Image upload
- Business Processes
  - Demo workflow routing, attachment, QPACS, data extraction, database integration
- Document Conversion
  - Demo PDF document conversion process

Five Things for You to Take Away

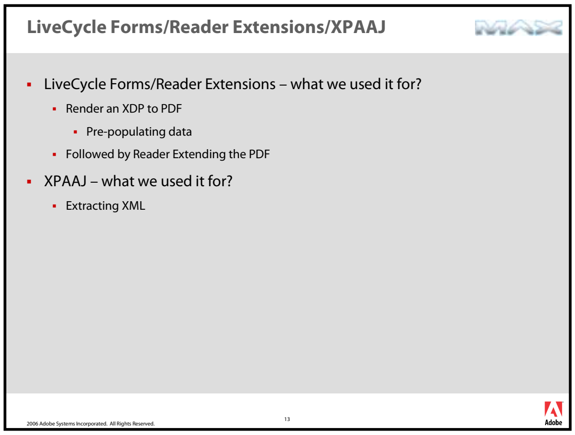
- Highlight of the 5 technologies used in the solution
  - Complex PDF form: Acrobat Dialog object, image field, dynamic forms, digital Signatures
  - Web Services with LiveCycle WF
  - LC Forms / XPAAJ library – when and how
  - Integration of Flex and LC WF and Flex
  - Integration of Flex UI and Adobe Reader – Frame, Session

Building a Complex Form using LiveCycle Form Designer

- Use data connections for external data sources
  - Web Services and OLE DB
- Use dynamic subforms – expand and shrink PDF forms
- Use Acrobat dialog object
- Use digital signature
  - Signature validation
  - Submit form to Workflow in PDF format to preserve the signatures
- Use image field

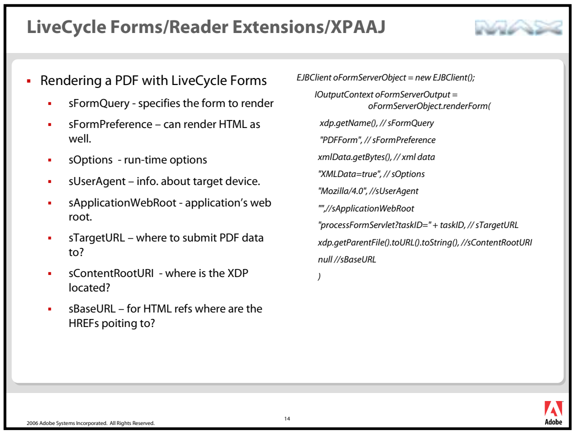
Web Services with LiveCycle Workflow

- Drag-Drop approach to creating a Web Service
- PDF calls a Web Service for validation, data lookups, etc.
  - Note: PDFs targeted for Adobe Reader requires Adobe Reader Extensions
- Quick Demo – "Hello World"



LiveCycle Forms/Reader Extensions/XPAAJ

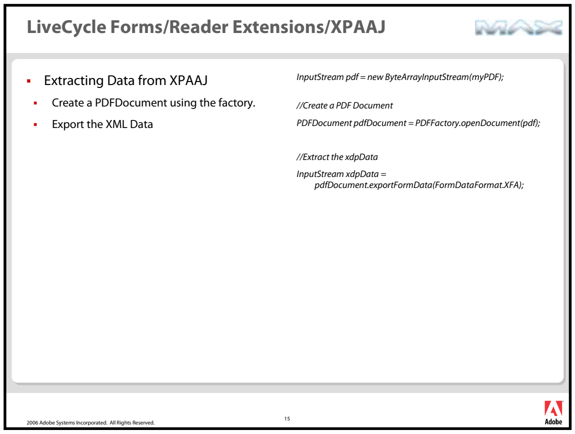
- LiveCycle Forms/Reader Extensions – what we used it for?
  - Render an XDP to PDF
    - Pre-populating data
  - Followed by Reader Extending the PDF
- XPAAJ – what we used it for?
  - Extracting XML



LiveCycle Forms/Reader Extensions/XPAAJ

- Rendering a PDF with LiveCycle Forms
  - sFormQuery - specifies the form to render
  - sFormPreference – can render HTML as well.
  - sOptions - run-time options
  - sUserAgent – info. about target device.
  - sApplicationWebRoot - application's web root.
  - sTargetURL – where to submit PDF data to?
  - sContentRootURI - where is the XDP located?
  - sBaseURL – for HTML refs where are the HREFs poiting to?

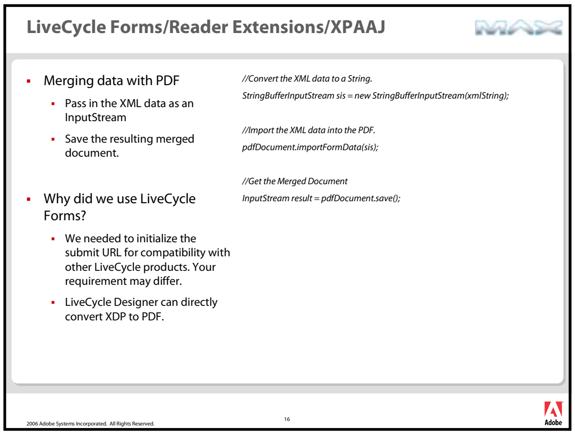
```
EJBClient oFormServerObject = new EJBClient();
    IOutputContext oFormServerOutput =
                oFormServerObject.renderForm(
     xdp.getName(), // sFormQuery
     "PDFForm", // sFormPreference
     xmlData.getBytes(), // xml data
     "XMLData=true", // sOptions
     "Mozilla/4.0", //sUserAgent
     "",//sApplicationWebRoot
     "processFormServlet?taskID=" + taskID, // sTargetURL
     xdp.getParentFile().toURL().toString(), //sContentRootURI
     null //sBaseURL
     )
```



LiveCycle Forms/Reader Extensions/XPAAJ

- Extracting Data from XPAAJ
- Create a PDFDocument using the factory.
- Export the XML Data

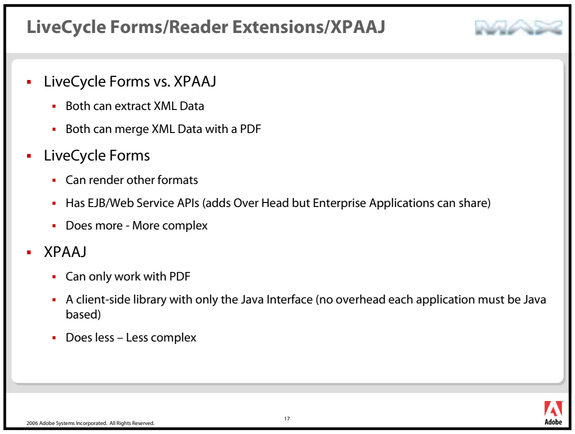
```
InputStream pdf = new ByteArrayInputStream(myPDF);

//Create a PDF Document
PDFDocument pdfDocument = PDFFactory.openDocument(pdf);


//Extract the xdpData
InputStream xdpData =
    pdfDocument.exportFormData(FormDataFormat.XFA);
```



LiveCycle Forms/Reader Extensions/XPAAJ

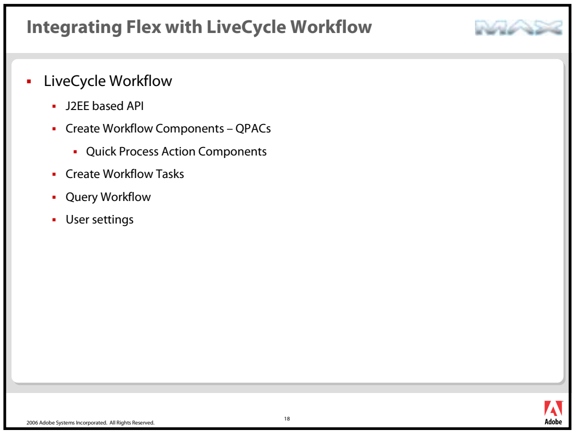
- Merging data with PDF
  - Pass in the XML data as an InputStream
  - Save the resulting merged document.
- Why did we use LiveCycle Forms?
  - We needed to initialize the submit URL for compatibility with other LiveCycle products. Your requirement may differ.
  - LiveCycle Designer can directly convert XDP to PDF.

```
//Convert the XML data to a String.
StringBufferInputStream sis = new StringBufferInputStream(xmlString);

//Import the XML data into the PDF.
pdfDocument.importFormData(sis);

//Get the Merged Document
InputStream result = pdfDocument.save();
```



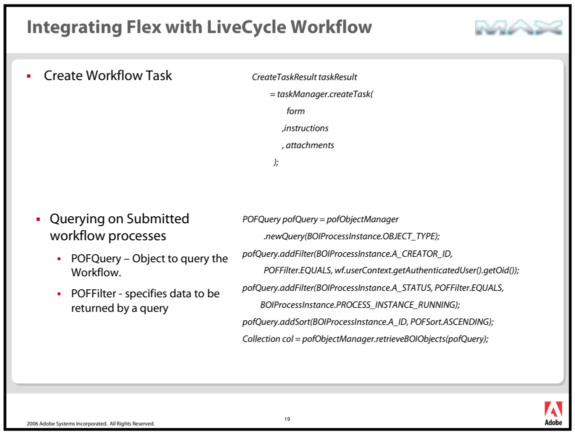
LiveCycle Forms/Reader Extensions/XPAAJ

- LiveCycle Forms vs. XPAAJ
  - Both can extract XML Data
  - Both can merge XML Data with a PDF
- LiveCycle Forms
  - Can render other formats
  - Has EJB/Web Service APIs (adds Over Head but Enterprise Applications can share)
  - Does more - More complex
- XPAAJ
  - Can only work with PDF
  - A client-side library with only the Java Interface (no overhead each application must be Java based)
  - Does less – Less complex



Integrating Flex with LiveCycle Workflow

- LiveCycle Workflow
  - J2EE based API
  - Create Workflow Components – QPACs
    - Quick Process Action Components
  - Create Workflow Tasks
  - Query Workflow
  - User settings



Integrating Flex with LiveCycle Workflow

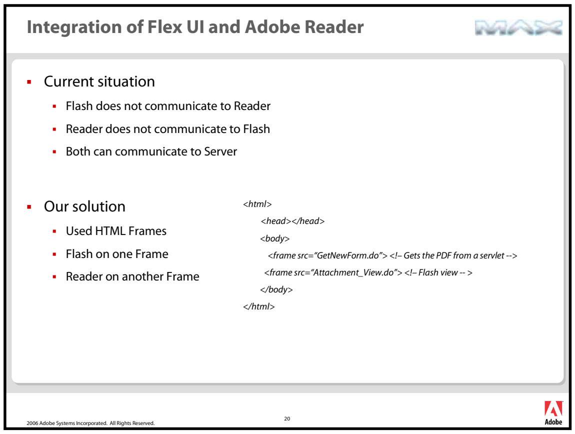
- Create Workflow Task

```
CreateTaskResult taskResult
        = taskManager.createTask(
            form
           ,instructions
           , attachments
        );
```

- Querying on Submitted workflow processes
  - POFQuery – Object to query the Workflow.
  - POFFilter - specifies data to be returned by a query

```
POFQuery pofQuery = pofObjectManager
        .newQuery(BOIProcessInstance.OBJECT_TYPE);
pofQuery.addFilter(BOIProcessInstance.A_CREATOR_ID,
        POFFilter.EQUALS, wf.userContext.getAuthenticatedUser().getOid());
pofQuery.addFilter(BOIProcessInstance.A_STATUS, POFFilter.EQUALS,
       BOIProcessInstance.PROCESS_INSTANCE_RUNNING);
pofQuery.addSort(BOIProcessInstance.A_ID, POFSort.ASCENDING);
Collection col = pofObjectManager.retrieveBOIObjects(pofQuery);
```

19



Integration of Flex UI and Adobe Reader

- Current situation
  - Flash does not communicate to Reader
  - Reader does not communicate to Flash
  - Both can communicate to Server

- Our solution
  - Used HTML Frames
  - Flash on one Frame
  - Reader on another Frame

```
<html>
     <head></head>
     <body>
       <frame src="GetNewForm.do"> <!– Gets the PDF from a servlet -->
      <frame src="Attachment_View.do"> <!– Flash view -- >
     </body>
<html>
```

20

Q&A

21

Better by Adobe™

22