Table of Contents

Introduction............................................................................................................ 6
Express Adaptor..................................................................................................... 6
Express Web Service Adaptor................................................................................ 6
Express Manager.................................................................................................... 7
Express Connector.................................................................................................. 7
Express ................................................................................................................ 7

Express Web Services Functional Overview .......................................................... 10

Using Express Web Services.................................................................................. 12
Express Web Services Architecture ........................................................................ 12
Single Server .................................................................................................... 12
Multiple Express Connectors ................................................................................ 12
Multiple Express Managers .................................................................................. 13
Multiple Express Adaptors ................................................................................... 14

Express Web Service Adaptor................................................................................ 15
Main Functions..................................................................................................... 15
Configuration....................................................................................................... 15
Operation ............................................................................................................ 15
AddJob ............................................................................................................. 15
AddJobFolder .................................................................................................... 17
DeleteJob ......................................................................................................... 18
DeleteJobFolder ................................................................................................. 19
EditJob ............................................................................................................. 19
GetConnectorStatusList ...................................................................................... 20
GetFileHash ...................................................................................................... 20
GetFileSize ....................................................................................................... 21
GetJobErrorDetail .............................................................................................. 21
GetJobFilesAsAttachment .................................................................................... 21
GetJobFilesAsStream .......................................................................................... 22
GetJobFilesMTOM .............................................................................................. 23
GetJobFileList .................................................................................................... 23
GetJobStatus .................................................................................................... 24
XML JobInfo Structure ....................................................................................... 25
Example #1 Web Service Adaptor Functions .......................................................... 27
Example #2 Web Service Adaptor Functions .......................................................... 28

Express Manager................................................................................................... 29
Express Manager Service ...................................................................................... 29
Express Manager User Interface ........................................................................... 29
Express Manager Menu ........................................................................................ 30
File Menu ......................................................................................................... 30
Tools Menu ....................................................................................................... 30
Help Menu ........................................................................................................ 30
Express Manager Options ...................................................................................... 31
Processing ........................................................................................................... 31
Express Connectors ............................................................................................. 33
Job Prioritization ................................................................................................. 34
Job Recovery ...................................................................................................... 34
List of Figures

Figure 1 - Express Web Services Software Architecture .................................................. 9
Figure 2 - Express Web Services Functional Overview ................................................... 11
Figure 3 - Multiple Express Web Services Connector Architecture ................................. 12
Figure 4 - Multiple Express Web Services Manager Architecture .................................... 13
Figure 5 - Multiple Express Web Services Architecture .................................................. 14
Figure 6 - XML Job Info Structure .............................................................................. 25
Figure 7 - Express Manager User Interface ................................................................... 30
Figure 8 - Express Manager Options ............................................................................ 31
Figure 9 - Express Manager eMail Notification ................................................................ 32
Figure 10 - Express Manager Options – Express Connectors ........................................... 34
Figure 11 - Express Connector User Interface .................................................................. 37
Figure 12 - Express Connector Options .......................................................................... 38
Figure 13 - Express Connector Options .......................................................................... 39
Figure 14 - Express Connector Logging .......................................................................... 40
Figure 15 - Express Connector Options – Express Manager .............................................. 41
Figure 16 – Sample XML Job Ticket File ...................................................................... 43
Figure 17 - Submit Problem to Adlib Screen ................................................................. 68
Introduction

Express Web Services is a suite of software applications that enable document processing requests (Jobs) to be initiated from different requestors, prioritized and distributed to one or more available Express Servers for processing.

Express Web Services provides scalability to Express Server so that multiple servers can work together to provide a unified document processing resource. Please refer to Figure 1 - Express Web Services Software Architecture. There are many benefits of this architecture:

**Scalability**

Express Servers may be combined together into a single document processing resource to increase document processing throughput. When additional document processing capacity is required, it is a simple matter of setting up a new Express Server and connecting it to the Express Web Services with no programming and minimal software configuration.

**Redundancy**

Combining Express Servers results into a single document processing resource provides N+1 redundancy. If one of the Express Server fails for any reason – hardware failure, software failure or network connectivity failure, then the other Express Servers will automatically assume the document processing load.

**Accessibility**

The Express Adaptor enables access to satisfy different interface requirements.

- The Express Web Service Adaptor will enable access from any computing platform on any network that has access to the Web Service.

- Future Adaptors may include:
  - Email Adaptor
  - ActiveX/.NET Adaptor
  - Watched Folder Adaptor

**Express Adaptor**

Express Web Services can accept document processing jobs from one or more Adaptors. The Adaptor is the interface between the Express Manager and external systems or users. The Express Web Service Adaptor is the first Adaptor to be released.

**Express Web Service Adaptor**

A Web Service is a collection of functions that are packaged as a single entity that is available for use by other programs over the Internet. Communication with a Web Service is done using standardized XML messaging. A Web Service by its nature is open and platform, operating system and application independent.

The Express Web Service Adaptor accepts new document processing jobs from a web service client using XML and SOAP messages. The interface for the Express Web Service can be queried from the Web Service and is described using the WSDL language.
The purpose of the Express Web Service Adaptor is to transfer incoming jobs to the Express Manager for processing. The Express Web Service Adaptor also enables jobs to be modified and deleted. Finally, the Express Web Service allows external applications to request the status of Jobs that are currently being processed.

A Job is made up of the following elements:

- **XML Job Ticket**: Defines the documents that are to be processed and the operations to be performed (i.e. Merge, OCR, Watermark, Header, Footer...) Refer to Express XML Job Tickets for more information.

- **XML JobInfo**: Defines how the Express Web Services should process the Job. (i.e. Job Priority, Job Schedule, ...) Refer to XML JobInfo Structure for more information.

- **Job Files**: Documents to be processed. Job Files can be referenced in the XML Job Ticket or in the XML JobInfo. When Job Files are referenced in the job ticket only, they are transferred from their location right into the Express Work directory. If the Job Files are referenced in the JobInfo XML, the files are transferred from the Express Manager into the Job Request Queue. This would relieve any rights issues within Express.

**Express Manager**

The Express Manager receives jobs from the Express Adaptor and manages the distribution of jobs to Express Servers for processing. The Express Manager maintains a list of all Express Servers that have been assigned to it. When a Job is to be processed, the Express Manager selects the first available Express Server and then sends the Job. The Express Manager tracks the status of each Job as it is processed by Express Server.

**Express Connector**

An Express Connector runs on each Express Server that is enabled to be used as part of the Express Web Services architecture. The Express Connector monitors the status of Express and receives Jobs from the Express Manager when Express is available for processing. Once a Job is received the Express Connector extracts the XML Job Ticket and Job Files and submits the Job to Express for processing.

**Express**

Express is a high performance server application that automates the conversion of your business content to Portable Document Format (PDF), Plain (ASCII) Text and Image files. Express converts an extensive list of file types including standard business documents, images, graphics, web and legacy files. In addition, it includes a comprehensive set of features such as Watermarking, PDF Security, PDF Optimization, PDF Form fill-in, PDF Merging, Splitting and Stitching into a single application. Express allows you to integrate document conversion into your workflow simply and affordably.
Express uses **Express XML Job Tickets** to specify how to process documents. The XML Job Ticket specifies the documents to convert, the document processing options, the output file types and destinations for processed documents.
Figure 1 - Express Web Services Software Architecture
Express Web Services Functional Overview

The following describe a typical sequence of events relating to processing a Job. Please refer to Figure 2 - Express Web Services Functional Overview for a diagram of these steps.

1. A Job, consisting of an XML Job Ticket and an XML JobInfo is submitted to the Express Web Service Adaptor using the AddJob() method.

2. The Express Web Service Adaptor transfers the Job to the Express Manager.

3. The Express Manager positions the Job in the JobInput Queue based on the Job Priority specified in the XML JobInfo.

4. The Express Manager locates the next available Express Server where the Job is to be processed.

5. The first Job in the JobInput Queue is sent to the Express Connector on the next available Express Server.

6. The Express Connector extracts the Job Ticket and optionally any Job Files and submits them to Express for processing.

7. When the Job is complete, Express transfers the processed documents (PDF/Image/Text Files) to the destinations defined in the XML Job Ticket.

8. The Express Connector reports the Job Status (Success or Failure) back to the Express Manager.

9. Job Status is reported back to the calling application via the Express Web Service Adaptor using the GetJobStatus() method.

10. The Job can be removed from Express Web Services by calling the DeleteJob() method of the Express Web Service Adaptor once the job is completed.
Figure 2 - Express Web Services Functional Overview

Adlib Express Web Services Functional Overview

Web Service Client

SOAP Message

XML Job Ticket

XML Job Info

Express Web Service Adaptor

- Add/Edit/Delete Jobs from Express Job Manager
- Get Job Status

Express Manager

Express Manager

MSMQ Message

XML Job Ticket

Queued Jobs

- Receives/Manages Jobs
- Sends Jobs to next available Server

Express Connector

- Receives/Manages Messages for Express Server
- Monitors Job Status

Express Server

- Receives XML Job Ticket
- Processes Documents
- Convert
- Publish
- Recognize

XML defines:
- Output format (e.g., PDF, HTML, TIF)
- Destination (folder, email, FTP, primit)

PDF

SERVER 1

SERVER 2

SERVER n
Using Express Web Services

Express Web Services Architecture

Express Web Services software components can be arranged in a very flexible architecture to suit a wide variety of application and document workflow requirements.

Single Server

For single server applications, all Express Web Services components and Express Server may be installed on a single server. This is advantageous when processing demand does not require more than one instance of Express Server. If processing demands increase to the point where a second Express Server is required then it is simple to add a second server that shares document processing with the initial server.

Multiple Express Connectors

For applications where a high volume of document processing is required or where redundancy is important, multiple Express Connectors with multiple Express Servers can be configured to operate as a single document processing resource from a single Express Manager. Refer to Figure 3 - Multiple Express Web Services Connector Architecture.
Multiple Express Managers

Express can be configured in an architecture where one or more Express Managers can share Express Connectors and Express Servers. This has the advantage of segregating Job processing of different workflows into different Express Managers while sharing a unified document processing resource. Refer to Figure 4 - Multiple Express Web Services Manager Architecture. In this example, Express Connector Servers 1 and 2 are shared between Express Manager Server 1 and 2. Express Connector Server n is dedicated to Express Manager Server 2.

Figure 4 - Multiple Express Web Services Manager Architecture
Multiple Express Adaptors

Express Manager can receive Jobs from one or more Express Adaptors. This is beneficial where applications may have more than one workflow process that needs to submit document processing Jobs. Multiple workflow processes or systems can share a unified document processing resource resulting in better utilization of available servers as well as lower maintenance costs. Refer to Figure 5 - Multiple Express Web Services Architecture.
Express Web Service Adaptor

Main Functions

The Express Web Service Adaptor provides an interface between Express Manager and any external entity to Add, Modify, and Delete Jobs. The Express Web Service Adaptor provides communication with Web Service Clients using SOAP Messages. The SOAP Message encapsulates the following information:

XML JobInfo
Stores parameters that shall be used to instruct Express Manager how to process the Job. Refer to XML JobInfo Structure for more details about how to define XML Job Info.

XML JobTicket
Stores a complete JobTicket which is passed on to the Express Server. Please refer to Express XML Job Tickets User’s Guide for more details about how to define XML Job Tickets.

JobFiles (Future)
Stores one or more files to be processed

Configuration

The exponentWSASettings.xml file must be modified to reference the Server Name where the Express Manager is running. Please refer to Express Web Service Adaptor installation instructions in the appendix for more details.

Operation

The Express Web Service Adaptor provides the following function calls (actual coding examples can be found in the installed Express WSA Sample Apps):

AddJob

Description:
Submits a Job to the Express Manager for processing.

Call:
AddJob(jobInfo, jobTicket)

jobInfo is a String representing an XML Job Info

jobTicket is a String representing an XML Job Ticket

Notes:
A Job may be submitted using UserJobID that must be specified in the JobInfo XML.

Returns:
XML JobInfo String updated with JobID.
**AddJobFilesAsAttachments**

Description:
Streams files via DIME attachment to the Express WSA Server

Call:
AddJobFilesAsAttachments(FileNames[], JobFolder, SpecificationType)

FileNames[] represents a String[] array that specifies the files that are streamed via DIME.

JobFolder represents a String that specifies the location where the files will be stored on the IIS server.

SpecificationType is DIME

Notes:
This method is only available when installing the Legacy Express WSA during Installation. If this is installed, the MTOM methodology is not available, and visa versa.

Returns:
String of XML data that specifies if the method was successful

**AddJobFilesAsStream**

Description:
Streams files via XML document to the Express WSA Server

Call:
AddJobFilesAsStream(JobFolder, XmlStream)

JobFolder represents a String that specifies the location where the files will be stored on the Express WSA Server.

XmlStream represents a String that has a structure similar to the following:
<JobFiles>
   <JobFile FileName="" Data="" Length=""/>
   ...
</JobFiles>

Notes:

Returns:
String of XML data that specifies if the method was successful
**AddJobFilesMTOM**

**Description:**
Streams files via MTOM byte arrays to the Express WSA Server. This method is flexible enough to allow for sending a large file in small chunks through a series of the same method call until the entire file has been sent.

**Call:**
```
AddJobFilesMTOM(FileName[], JobFolder, buffer[][], Offset[], BytesRead[])
```

- FileName[] represents a String[] array that specifies the file names of the file bytes that are streamed via MTOM.
- JobFolder represents a String that specifies the location where the files will be stored on the IIS server.
- buffer[] represents the array of byte arrays which are the file bytes.
- Offset[] represents an array of each file’s offset when sending file in chunks (i.e. the zero-based position of the where to continue adding the bytes). If it is the first call or the entire file is being sent, the value is 0.
- BytesRead represents an array of each file’s maximum number of bytes to read/write (usually the size of the buffer).

**Notes:**
This method is only available when installing the Default Express WSA during Installation. If this is installed, the DIME methodology is not available, and visa versa.

This method is very similar to the logic used when calling a FileStream.Write() command in C#, and the FileOutputStream.write() Java, etc.

**Returns:**
String of XML data that specifies if the method was successful

---

**AddJobFolder**

**Description:**
Creates a unique Job Folder on the IIS Web Server where the Express Web Service Adaptor is running. The AddJobFolder returns a string that represents the path of the Job Folder. The purpose of this folder is for Job Files to be uploaded prior to processing as part of a specific job. The Job Ticket must refer to the files in the JobFolder so that they can be accessed by Express. Refer to **Submitting Files using AddJobFolder** for a detailed description of how to use this method.

**Call:**
```
AddJobFolder()
```

**Notes:**

Returns:

String of XML that represents an Input Folder and an Output Folder. The following is a sample of the XML String that is returned using the AddJobFolder method.

```
<FolderInfo FolderID='7923c558d8d-e2e9-495d-9504-3cfe4ad86d55'>
  <InputFolder>
    http://MySvr/exponentJobs/7923c558d8d-e2e9-495d-9504-3cfe4ad86d55/Input/
  </InputFolder>
  <OutputFolder>
    http://MySvr/exponentJobs/7923c558d8d-e2e9-495d-9504-3cfe4ad86d55/Output/
  </OutputFolder>
</FolderInfo>
```

**DeleteJob**

Description:

Deletes the specified Job from within the Express Manager.

Call:

`DeleteJob(jobID,isUserJobID)`

`jobID` is a String representing the JobID or the User JobID

`isUserJobID` is a boolean indicating if the jobID string represents the JobID or the User JobID. A True value indicates the User JobID.

Notes:

If DeleteJob is invoked and the Job has been completed then the Job Message is removed from the Job Complete Queue.

Returns:

Boolean
**DeleteJobFolder**

Description:
Deletes the folder(and files, if they exist) specified located on the Express WSA Server

Call:
DeleteJobFolder(JobFolder)

JobFolder is a String of the name of the folder that exists on the IIS server. Ie. DeleteJobFolder ("1129200411_36_09_728a32fcc15-8621-48eb-8ff4-335f1f15e5ea")

Notes:

Returns:
Xml to indicate if delete was successful

**EditJob**

Description:
Sends an updated XML JobInfo to the Express Manager for processing. If the Job referenced in the XML JobInfo has not been sent to a Express Adaptor for processing, then the XML JobInfo shall be updated.

Call:
EditJob(jobInfo)

jobInfo is a String representing an XML Job Info

Notes:
This method shall initially support modifying the Job Priority only.

The XML Job Info must contain the JobID and optionally the UserJobID in order to successfully locate and update the Job.

Returns:
XML Job Info String
**GetConnectorStatusList**

**Description:**
Returns the list of Connectors in the Manager’s AdlibServerList.xml

**Call:**
GetConnectorStatusList

**Notes:**
This method will return the Connector list that the Manager is using for job allocation. The server list will usually contain a status of the Connectors. If a status is not present then a Connector is being initialized in the Manager’s list.

**Returns:**
AdlibServerList.xml

**GetFileHash**

**Description:**
Returns the checksum hash string of the specified file.

**Call:**
GetFileHash(Filename, JobFolder, HashType)

Filename represents the name of the file in question

JobFolder represents a String that specifies the location where the files will be stored on the IIS server.

HashType is either MD5 or SHA1 (depending on the type of cryptography hash required)

**Notes:**

**Returns:**
String containing the hash of the file.
**GetFileSize**

**Description:**
Returns the size (in bytes) of the specified file in the specified job folder.

**Call:**
GetFileSize(Filename, JobFolder)

- Filename represents the name of the file in question
- JobFolder represents a String that specifies the location where the files will be stored on the IIS server.

**Notes:**

**Returns:**
A number which represents the size of the file (in bytes).

**GetJobErrorDetail**

**Description:**
Returns detailed information regarding the job id specified.

**Call:**
GetJobErrorDetail(jobID,isUserJobID)

- jobID is a String representing the JobID or the User JobID
-.isUserJobID is a boolean indicating if the jobID string represents the JobID or the User JobID. A True value indicates the User JobID.

**Notes:**

**Returns:**
XML containing detailed log information regarding the job from Express/Connector and Manager.

**GetJobFilesAsAttachment**

**Description:**
Returns the specified file(s) as a DIME attachment

**Call:**
GetJobFilesAsAttachment(JobFolderName, JobFileName, Specification)

- JobFolderName is a String of the name of the folder that exists on the IIS server. I.e. “http://localhost/exponentjobs/1129200411_36_09_728a32fcc15-8621-48eb-8ff4-335f1f15e5ea/Input”
- JobFileName is a string representing the filename to look for in the location specified.
Ie. “test.doc”

If JobFileName is null, then the method will return all the files in the folder specified.

SpecificationType is DIME

Notes:
This method is only available when installing the Legacy Express WSA during Installation. If this is installed, the MTOM methodology is not available, and visa versa.

Returns:
DIME attachment(s) that is streamed through the webservice.

**GetJobFilesAsStream**

Description:
Returns the specified file(s) as a XML document

Call:
GetJobFilesAsStream(JobFolderName, JobFileName)

JobFolderName is a String of the name of the folder that exists on the IIS server. Ie. “http://localhost/exponentjobs/1129200411_36_09_728a32fcc15-8621-48eb-8ff4-335f1f15e5ea/Input”

JobFileName is a string representing the filename to look for in the location specified. Ie. “test.doc”

If JobFileName is null, then the method will return all the files in the folder specified.

Notes:
XML Structure to come back from the webservice will look similar to the following example:

```xml
<JobFiles>
    <JobFile FileName="" Data="" Length=""/>
    ...
</JobFiles>
```

The value enclosed in the Data attribute is Base64 encoded data.

Returns:
XML document that is streamed through the webservice.
GetJobFilesMTOM

Description:
Returns the specified file(s) as MTOM byte arrays

Call:
GetJobFilesMTOM(FileName[], JobFolder, Offset[], BufferSize[])

FileName[] represents a String[] array that specifies the file names of the file bytes that are streamed via MTOM.

JobFolder is a String of the name of the folder that exists on the IIS server. Ie. “http://localhost/exponentjobs/1129200411_36_09_728a32fcc15-8621-48eb-8ff4-335f1f15e5ea/Output”

Offset[] represents an array of each file’s offset when receiving the file in chunks (i.e. the zero-based position of the where to continue adding the bytes). If it is the first call or the entire file is being sent, the value is 0.

BufferSize[] represents how many bytes to be sent back in the byte array(s). If getting the entire file at once, this can be set to the filesize (which is known by calling the GetFileSize method).

Notes:
If FileName[] array is null, then the method will return all the files in the folder specified.

This method is only available when installing the Legacy Express WSA during Installation. If this is installed, the MTOM methodology is not available, and visa versa.

Returns:
byte arrays (the file data) that are streamed through the webservice.

GetJobFileList

Description:
Returns the list of files store in the specified location

Call:
GetJobFileList(JobFolder)

JobFolder represents a String of the folder name that exists on the Express WSA Server.
Ie. GetJobFileList(“http://localhost/exponentjobs/1129200411_36_09_728a32fcc15-8621-48eb-8ff4-335f1f15e5ea/Output”)

Notes:

Returns:
String of all the files located in the JobFolder specified.

**GetJobPosition:**

**Description:**
Returns the position of the job in the current queue

**Call:**
GetJobPosition(jobID)

jobID is a String representing the JobID or the User JobID

**Notes:**

**Returns:**
String value that represents the current position of the job in the queue

**GetJobStatus**

**Description:**
Returns the Job Status of the specified Job.

**Call:**
GetJobStatus(jobID,isUserJobID)

jobID is a String representing the JobID or the User JobID

isUserJobID is a boolean indicating if the jobID string represents the JobID or the User JobID. A True value indicates the User JobID.

**Notes:**

**Returns:**
XML JobInfo updated with current Status.
XML JobInfo Structure

The XML Job Info structure stores information that is used to instruct Express Manager how to process the Job. (see Figure 6 –XML Job Info Structure).

Figure 6 –XML Job Info Structure

```xml
<JobInfo>
    <JobSettings JobID="" UserJobID="" JobPriority="" AdlibServer="" AdlibServerGroup=""/>
    <JobSchedule StartDate="YYYYMMDD" StartTime= "HH:MM:SS.sss" />
    <JobFileMsg Count=""/>
    <JobFileList>
        <JobFile Filename="" Folder=""/>
        <JobFile Filename="" Folder=""/>
        <JobFile Filename="" Folder=""/>
    </JobFileList>
    <CurrentJobStatus Status="Pending" ConnectorStatus="Online" />
    <exponentManager ServerName="/">
    <exponentConnector ServerName="/">
    <JobStatus Status= ""/>
    <JobStatus History>
        <JobStatus Date= "YYYYMMDD" Time= "HH:MM:SS.sss" Status=""/>
        <JobStatus Date= "YYYYMMDD" Time= "HH:MM:SS.sss" Status=""/>
        <JobStatus Date= "YYYYMMDD" Time= "HH:MM:SS.sss" Status=""/>
    </JobStatus History>
</JobInfo>
```

Notes:

JobID is populated when the Job is sent to the Express Manager.

UserJobID is optionally used to define a identifier that the end user requires.

JobPriority is a numeric value from 1 to 5. (1 is the highest and 5 is the lowest.)

AdlibServer is a string value indicating which Connector to use for job processing.

AdlibServerGroup is a string value indicating which group of Connectors to use for job processing.

JobSchedule determines when the Job is to be processed. (Future)

JobFileMsg is reserved for internal application use only.

JobFile is used to specify files which are to be transferred to Express by Express Web Services.

JobStatus is the current status of the job as it is processed by Express Web Services.
CurrentJobStatus is used to identify the current status of the job as well as whether or not there are any connectors online to process this job. If the ConnectListStatus is Offline then there are no Connectors online to process this job.

Examples of Job Statuses are:

1. Pending
2. InProgress
3. Completed
4. Failed

JobStatusHistory is populated when the current status of the job has been updated. The status can be obtained from the GetJobStatus method from the Express Web Service.
**Example #1 Web Service Adaptor Functions**

To submit a Job to Express Web Services for processing using the Express Web Service Adaptor you must call the AddJob function as follows:

```csharp
AddJob(string JobInfo, string JobTicket);
```

where JobInfo is the following string of XML:

```xml
<JobInfo>
    <JobSettings JobID="" UserJobID="" JobPriority="3" AdlibServer="" AdlibServerGroup="" />
    <JobFileMsg Count="" />
    <JobFileList>
        <JobFile Folder="" FileName="" />
    </JobFileList>
</JobInfo>
```

and JobTicket is the following string of XML:

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<?AdlibExpress applanguage="USA" appversion="2.5" dtdversion="1.0" ?>
<!DOCTYPE JOBS SYSTEM "C:\Adlib Express\DTD\AdlibExpress.dtd">
    <JOB>
        <JOB:DOCINPUTS>
            <JOB:DOCINPUT FILENAME="test.doc" FOLDER="\localhost\C" />
            <JOB:DOCINPUT FILENAME="excel.xls" FOLDER="\localhost\C" />
        </JOB:DOCINPUTS>
        <JOB:DOCOUTPUTS>
            <JOB:DOCOUTPUT FILENAME="test.pdf" FOLDER="\localhost\C" DOCTYPE="PDF" />
        </JOB:DOCOUTPUTS>
        <JOB:SETTINGS>
            <JOB:HEADER ENABLED="Yes" TEXTRIGHT="&amp;[Page] of &amp;[Pages]" LAYER="Foreground" />
            <JOB:WATERMARK ENABLED="Yes" FONTCOLOR="6447714" FONTNAME="Helvetica" FONTSIZE="48" LAYER="Foreground" TEXT="TESTING!" />
        </JOB:SETTINGS>
    </JOB>
</JOBS>
```

The AddJob method will return the XML JobInfo updated with the JobID as follows:

```xml
<JobInfo>
    <JobSettings JobID="c4ac36cd-0375-4fe1-bd35-45f80a9cf319\20327" UserJobID="" JobPriority="3" AdlibServer="" />
    <JobFileMsg Count="" />
    <JobFileList>
        <JobFile Folder="" FileName="" />
    </JobFileList>
</JobInfo>
```

The AddJob method will return the XML JobInfo updated with the JobID as follows:
Example #2 Web Service Adaptor Functions

To specify a connector to process your jobs you can modify the JobInfo xml to the following:

```xml
<JobInfo>
  <JobSettings JobID="" UserJobID="" JobPriority="3" AdlibServer="YOURSERVERNAME" />
  <JobFileMsg Count="" />
  <JobFileList>
    <JobFile Folder="" FileName="" />
  </JobFileList>
</JobInfo>
```

To specify a group of the connectors to process your jobs, modify the JobInfo xml to the following:

```xml
<JobInfo>
  <JobSettings JobID="" UserJobID="" JobPriority="3" AdlibServer=""
AdlibServerGroup="YOURSERVERGROUPNAME" />
  <JobFileMsg Count="" />
  <JobFileList>
    <JobFile Folder="" FileName="" />
  </JobFileList>
</JobInfo>
```

To make a connector part of a particular group look refer to Express Connectors.
Express Manager

The Express Manager software is comprised of two components: the Express Manager Service and the Express Manager User Interface. Express Manager is responsible for the following functions:

- Accept and Prioritize Incoming Jobs from Express Adaptors
- Maintain List of Express Connectors that are available for Job Processing.
- Transfer Jobs to available Express Connectors.
- Track Job Status.
- Recover and Reprocess failed Jobs.

Express Manager Service

The Express Manager Service is responsible for all Job processing and monitoring functions. The Express Manager Service may be accessed from the Services control panel from within Administrative Tools.

To open the Services control panel:
- From Windows NT, click Start, point to Settings, click on Control Panel, and then double-click Services
- From Windows 2000, click Start, point to Settings, and then click Control Panel. Double-click Administrative Tools, and then double-click Services

Express Manager User Interface

The Express Manager User Interface screen (see Figure 7 - Express Manager User Interface) provides controls to Start and Stop the Express Manager from monitoring connectors as well as providing access to Options screens. It also includes feedback on the status of Express Manager. The Express Manager User Interface screen does not need to be running for Express Manager Service to operate.

The Express Manager starts processing Jobs by clicking on the Start Monitoring button. The Express Manager will stop processing Jobs by clicking on the Stop Monitoring button. Jobs will still be received from any Express Adaptor provided the Express Manager Service is running. These Jobs will only be processed once the Start Monitoring button is clicked.
Express Manager Menu

The Express Manager menu items are described below.

File Menu

The **Save Express Web Services Settings** allows users to save all Express Web Services settings to an XML file. The **Load Express Web Services Settings** Menu Item allows users to load Express Web Services settings from an XML file. This allows users to backup and load the Express Web Services settings on other computers.

Tools Menu

The **Start Monitoring/Stop Monitoring** item allows users to start and stop monitoring for Jobs. It also provides access to the **Event Logs** and **Express Manager Options**.

Help Menu

The Express Manager Help Menu provides access to the Express Web Services User Guide, the Express XML Job Ticket User Guide as well as the Help About for Express Connector. It also includes the **Submit a Problem** item that simplifies and automates the problem reporting process to Adlib.
Express Manager Options

Express Manager contains several screens for configuring the application to meet specific needs. The Options screen is accessible from the Options button on the main screen or by selecting the Express Manager Options item from the Tools menu or by pressing the F2 function key. Each of the Options screens is described in detail below. (see Figure 8 - Express Manager Options).

Processing

Enable Event Log

Express Manager can create a log of events as Jobs are processed.

The log file is stored in XML format and contains the following items:
- Time
- Date
- Description
- Thread Name
The **Log Mode** defines the level of detail that is included in the log. The **Log Folder**, defined by the user specifies where the Log is stored. The **Retention Period** defines how long the log data is maintained in the log file. The **Maximum File Size** defines how large the log file can grow to prior to discarding the oldest log data. If **Retain Log History** is enabled then all log data is maintained in archive files rather than being discarded as the Retention Period or Maximum File Size is exceeded. The user may optionally compress the archived log data using the zip compression algorithm in order to save disk space. The **eMail Express Manager** problem enables Manager to send emails to a defined destination when a job fails, or a connector goes offline.

**Figure 9 - Express Manager eMail Notification**

Select either Outbox or SMTP to send emails when a Job fails or a connector goes offline. Simply fill in the From and To textboxes. If a job fails, an eMail with detailed information
about the failure. If a connector goes offline, then an eMail will be sent that outlines the reason why the connector went offline.

*Note: When using this feature specify the from address as the machine submitting the email (localhost@domain.com). This will make it easier to figure out which machine went offline/failed your job.

**Express Connectors**

Express Manager can issue Jobs to different Express Connectors. When an Express Connector is configured to accept Jobs from an Express Manager, the Express Connector sends a message to the Express Manager to update the list of Express Connectors and enable the server to be able to receive Jobs. (see Figure 10 - Express Manager Options – Express Connectors).

The **Computer Name** column displays all servers you have defined as having an Express Connector. The **Enabled** column indicates the Express Connector Servers that are currently enabled to receive Jobs from the Express Manager.

To define an Express Manager, click on the **Add** button and type in the server name of the desired Express Manager. Note that the Express Connector must be configured to receive Jobs from the Express Manager.

To remove an Express Connector from the list, click on the **Delete** button.

To define a group name for a connector make sure you assign it under the **Group Name** column. Once this is defined you can modify your jobinfo xml to look like this example:

```
<JobInfo>
  <JobSettings JobID="" UserJobID="" JobPriority="3" AdlibServer="" AdlibServerGroup="YOURSERVERGROUPNAME" />
  <JobFileMsg Count="" />
  <JobFileList>
    <JobFile Folder="" FileName="" />
  </JobFileList>
</JobInfo>
```

The group name column is helpful in situations where only a certain number of servers can perform processing options such as OCR. You can define a group of servers as an OCR group. This will allow you to programatically define OCR jobs to goto that particular server group.
Job Prioritization

Jobs submitted to Express Manager have a priority assigned to them in the Job Info XML. The Priority is a numeric value that ranges from 1 to 5. 1 is the highest priority and 5 is the lowest priority. Express Manager sorts incoming Jobs based on their priority and then by the time they were submitted to Express Manager for processing. All Jobs that have a priority of 1 will be processed before any Jobs that have a priority of 2 and so on. Within a particular priority level, the oldest Job is processed first. If low priority Jobs are not getting processed by Express Manager, it is due to the fact that higher priority Jobs are arriving and are getting processed first. The remedy for this problem is to increase the number of Express Server processing Jobs or to increase the priority level of the lowest priority Jobs to ensure that they get processed.

Job Recovery

Express Manager incorporates a Job Recovery scheme to ensure that all Jobs get processed even if one of the Express Servers fails or becomes unavailable for Job processing. Express
Manager maintains a list of all Jobs that have been sent to one or more Express Connectors for processing. If a Job remains in that list longer than a preset timeout value, then Express Manager automatically requests the Job Status from the Express Connector where the Job was sent. If the Job Status indicates that the Job is still being processed, then the timeout value is reset. If no Job Status is reported by Express Connector, then Express Manager assumes that the Express Server is unavailable. The Job is then resubmitted to the JobInput Queue with the highest priority and processed by the next available Express Connector.
Express Connector

The Express Connector software is comprised of two components: the Express Connector Service and the Express Connector User Interface. Express Connector is responsible for the following functions:

- Accept Jobs from Express Manager.
- Extract XML Job Ticket and JobFiles and submit to Express for processing.
- Monitor and Report Job Status of Jobs being processed.

Express Connector Service

The Express Connector Service is responsible for all Job processing and monitoring functions. The Express Connector Service may be accessed from the Services control panel from within Administrative Tools.

To open the Services control panel:
- From Windows NT, click Start, point to Settings, click on Control Panel, and then double-click Services
- From Windows 2000, click Start, point to Settings, and then click Control Panel. Double-click Administrative Tools, and then double-click Services

Express Connector User Interface

The Express Connector User Interface screen (see Figure 11 - Express Connector User Interface) provides controls to Start and Stop the Express Connector Service as well as providing access to Options screens. It also includes feedback on the status of both Express Connector and Express Server. The Express Connector User Interface screen does not need to be running for Express Connector Service to operate.

The Express Connector Service is started by clicking on the Start button. After starting, Express Connector will send a notification message to the Express Manager Server(s) that are configured for use with the Express Connector. The Express Manager will then send Jobs to the Express Connector for processing.

Express Connector Service can be stopped by clicking on the Stop button. Jobs will no longer be accepted for processing while the Express Connector Service is stopped.
Express Connector Menu

The Express Connector menu items are described below.

**File Menu**

The **Save Express Web Services Settings** allows users to save all Express Web Services settings to an XML file. The **Load Express Web Services Settings** Menu Item allows users to load Express Web Services settings from an XML file. This allows users to backup and load the Express Web Services settings on other computers.

**Tools Menu**

The **Start/Stop** item allows users to start and stop the Express Connector Service. It also provides access to the **Event Logs** and **Express Connector Options**.

**Help Menu**

The Express Connector Help Menu provides access to the Express Web Services User Guide, the Express XML Job Ticket User Guide as well as the Help About for Express Connector. It also includes the **Submit a Problem** item that simplifies and automates the problem reporting process to Adlib.
Express Connector Options

Express Connector contains several screens for configuring the application to meet specific needs. The Options screen is accessible from the Options button on the main screen or by selecting the Express Connector Options item from the Tools menu or by pressing the F2 function key. Each of the Options screens is described in detail below.

Processing

Maxmium Jobs

Figure 12 - Express Connector Options

Maxmium Jobs sets the total number of Job Tickets that are allowed to be queued by the Express Connector. You can specify a range from 1-25 queued Job Tickets.

Setting this value to 1 means that Express Manager will only send 1 Job to Express Connector and will wait until that Job is completed prior to sending the next Job. This ensures that Jobs are strictly processed in the order that they are received by Express Manager, however, the overhead of communicating Jobs from Express Manager to Express Connector will result in overall slower document processing performance.

Setting this value greater than 1 will mean that Express Manager will send more than one Job to Express Connector for processing at one time. This ensures that Express processes Jobs continuously. This results in maximum document processing performance, but may mean that Jobs are not processed strictly in the order that they are received.
Express Option Configuration

Express allows you to setup Multiple Option Configurations for document conversion. Each Option Configuration stores a different profile, each with its own set of Folders, Processing, Scheduler, PDF Options, Print Options and Branding Options. This feature allows users to easily change from one configuration to another without having to change the Options Settings.

A valid Express Option Configuration must be selected in order for Express Connector to process job tickets. If you wish to add configurations to this screen, please see the Express configuration setup located in the Express User’s Guide.

Express Server must be installed and configured correctly in order to function properly with the Express Connector. Please refer to Express Server Configuration in the Appendix for details.

Figure 13 - Express Connector Options

Make configuration compatible button will modify the selected configuration to be compatible with Express Connector’s requirements. You must close Express Server or restart the Express Server service after this action in order to apply the changes.

Copy URI file(s) locally will allow connector to download any required files specified in the job ticket and place them in the Express work folder before submitting the job. This allows
Express server to concentrate on conversion and to leave the downloading/copying of the file up to connector. The ExponentConnectorSettings.xml file also contains an attribute called CopyURIThreads. This setting allows the Connector to use multithreading to download the required files. The default is one thread to download the required files. The maximum recommended threads is five.

Enable Logging

Express Connector can create a log of events as Jobs are processed.

Figure 14 - Express Connector Logging

The log file is stored in XML format and contains the following items:
- Time
- Date
- Description
- Thread Name
- Thread Instance
- Error Number
- Error Description
- Server Name
- Queue Name
- Job Id
- Msg ID

The Log Mode defines the level of detail that is included in the log. The Log Folder, defined by the user specifies where the Log is stored. The Retention Period defines how long the log data is maintained in the log file. The Maximum File Size defines how large the log file can grow to prior to discarding the oldest log data. If Retain Log History is
enabled then all log data is maintain in archive files rather than being discarded as the Retention Period or Maximum File Size is exceeded. The user may optionally compress the archived log data using the zip compression algorithm in order to save disk space.

**Express Managers**

Express Connector can accept Jobs from different Express Managers. Express Connector must be configured with the computer name(s) where Express Manager resides.

The **Computer Name** column displays all servers you have defined as having an Express Manager. The **Enabled** column indicates the Express Manager Servers that are currently enabled to send Jobs to the Express Connector.

To define an Express Manager, click on the **Add** button and type in the server name of the desired Express Manager.

To remove an Express Manager from the list, click on the **Delete** button.

**Figure 15 - Express Connector Options – Express Manager**
Express XML Job Tickets

An XML Job Ticket is an electronic file that contains all of the information required for the production of a document processing job. The Job Ticket is a set of specific document processing options used by Express to process the Job. The XML file format uses a plain (ASCII) text file with an .XML extension and uses Extensible Markup Language (XML) standards to store the job instructions. Please visit the World Wide Web Consortium (W3C) web site at www.w3c.org/XML for additional information on XML.

The XML and XML Job Tickets support all of the features provided by the Express user interface, such as the following:

- Header and Footer
- Watermark
- PDF Overlay
- PDF Bookmarks
- OCR
- PDF Splitting/Merging/Stitching
- PDF Page and Content Scaling
- PDF Settings and Print Settings
- PDF Document Information
- Document Open Options
- CAD, HTML, Text Page Setup

In addition to these features, XML/XML Job Tickets extend the functionality of the Express User Interface, including the following advanced features:

- File Merging
- Table of Contents Report
- Bookmark Creation
- Printing and FTP of processed documents
- PDF Information Extraction

XML Job Tickets allow you to specify the document(s) that are to be processed within the XML structure. The documents to be processed may be referenced using a UNC path or a URI (URL, FTP, HTTP).

XML Job Tickets allow you to specify the following destinations for the processed documents:

- Network Folders/Paths
- eMail Addresses
- Printers
- URI (URL, FTP, HTTP)
<?xml version="1.0" encoding="ISO-8859-1" ?>
<AdlibExpress applanguage="USA" appversion="2.8.0" dtdversion="1.9.4" />
<!DOCTYPE JOBS SYSTEM "C:\Adlib Express\DTD\AdlibExpress.dtd">
  <JOB:DOCINPUTS>
    <JOB:DOCINPUT FILENAME="Water on the Space Station.doc" FOLDER="C:\Adlib Express Job Tickets\Docs" />
  </JOB:DOCINPUTS>
  <JOB:DOCOUTPUTS>
    <JOB:DOCOUTPUT FILENAME="Defaults.pdf" FOLDER="C:\Adlib Express Job Tickets\XML Job Tickets Output\" />
  </JOB:DOCOUTPUTS>
  <JOB:DESTINATION>
    <JOB:DESTFOLDERS />
    <JOB:DESTPRINTERS />
    <JOB:DESTEMAILS />
    <JOB:DESTURIS />
  </JOB:DESTINATION>
  <JOB:LOGSETTINGS />
  <JOB:SCRIPTSETTINGS />
  <JOB:SETTINGS>
    <JOB:BATESNUMBERING />
    <JOB:CADD CONVERSION />
    <JOB:FOOTER />
    <JOB:HEADER />
    <JOB:HTMLCONVERSION />
    <JOB:IMAGESETTINGS />
    <JOB:METADATEAELEMENTS>
      <JOB:METADATEAELEMENT />
    </JOB:METADATEAELEMENTS>
    <JOB:NATIVEAPPSETTINGS>
      <JOB:MSPOWERPOINT />
    </JOB:NATIVEAPPSETTINGS>
    <JOB:OCRSSETTINGS>
      <JOB:IMAGERECONVERSION />
      <JOB:PDFPROCESSING />
      <JOB:DOCUMENTPROCESSING />
      <JOB:OCRLANGUAGES />
      <JOB:OCRLOGSETTINGS />
    </JOB:OCRSSETTINGS>
  </JOB:SETTINGS>
</JOB>
Document Type Definition (DTD)

XML Job Ticket files do not use any of the Express settings defined in the User Interface because the Express document type definition (DTD) (AdlibExpress.dtd) referenced in the XML Job Ticket file defines the settings used and their defaults. The AdlibExpress.dtd file resides in the DTD folder of the Express installation folder (e.g. C:\Adlib Express\DTD).

The DTD is used to validate the XML Job Ticket. The XML Job Ticket is **valid** if it complies with the constraints expressed in the DTD. Express validates the XML Job Ticket prior to processing it. Users can validate the XML Job Ticket structure by opening it with IE 5.0 or greater. A plug-in tool for IE 5.0 and up can be downloaded and used to validate both the structure and syntax of an XML file by using the right click tools menu. It can be downloaded [here](#).

**Note:** If you are using Express 3.0.0 or greater than you can use the default Express DTD. This is usually located on the connector machine under the following folder; $SYSTEMROOT\Adlib Express\DTD

**DTD Default Settings**

If you wish to modify the default settings in the AdlibExpress.dtd, you must copy the AdlibExpress.dtd and give it a new filename. The new DTD must be referenced in your XML Job Ticket. Note that if you should not modify anything other than the default settings. The only time when these DTD defaults are used, is when you have a tag of the same name inside your XML Job Ticket. The defaults will be loaded for that tag alone but will be overwritten by any specific settings inside your XML file. If you wish to take advantage of all the default settings of your modified DTD, then a skeleton tag is required for each section you wish to have default settings applied to.

For more details about XML Job Tickets and DTDs, please refer to the Express XML Job Ticket User’s Guide.
Submitting Files to Express Web Services for Processing

There are a variety of methods that can be used to submit files via Express Web Services to Express Server for processing. The selection of these methods depends on your workflow, the location of the files to be processed, and network security.

Submitting Files via Job Ticket Reference

Users can submit files to Express Web Services for processing by referring to the files in an XML Job Ticket. Express Manager will receive the Job Ticket from the Adaptor and then pass the Job Ticket to Express Connector. Express Connector will submit the Job Ticket to Express for processing. The files referred to in the Job Ticket will be accessed by Express so that they can be processed.

To submit files via Job Ticket, an XML Job Ticket must be created that has a DocInput defined for each file to be processed as part of the Job Ticket. DocInputs may be defined using the following protocols:

UNC Path

The following is a snippet of an XML Job Ticket file showing an UNC Path:

```xml
<JOB:DOCINPUTS>
  <JOB:DOCINPUT FILENAME="test.doc" FOLDER="\localhost\C\" />
  <JOB:DOCINPUT FILENAME="test.xls" FOLDER="\myserver\fileshare\" />
</JOB:DOCINPUTS>
```

File access rights must be configured such that each server running Express has rights to access the files in the specified folder. Note that if no folder is specified, then the file is assumed to be in the same folder as the Job Ticket.

URI

The following is a snippet of an XML Job Ticket file showing an HTTP URI:

```xml
<JOB:DOCINPUTS>
</JOB:DOCINPUTS>
```

The URIDOWNLOAD tag specifies if the file is to be downloaded locally prior to processing.

The following is a snippet of an XML Job Ticket file showing an FTP URI:

```xml
<JOB:DOCINPUTS>
  <JOB:DOCINPUT URI="ftp://username:password@www.myserver.com:21/exponentJobs/myFile.doc" URIDOWNLOAD="Yes" URITIMEOUT="60"/>
</JOB:DOCINPUTS>
```

Submitting Files via JobInfo Reference

Users can also submit files to Express Web Services directly by referring to the files in the XML Job Info string that is passed to Express Manager via the Express Web Service Adaptor. In this case, Express Manager will access the files and copy them to the Express Manager Server. The files are then compressed and submitted to Express Connector where they will...
be uncompressed prior to submitting the Job to Express for processing. This method of submitting files may be advantageous when it is difficult to ensure that all Express Servers will have sufficient rights to access the files in their original location. Please note that the Express Manager Server must have access rights to the files.

When submitting files via Express Web Services, the Job Ticket must be constructed to refer to the file as a DocInput, however, the folder attribute does not need to be defined. The following is a snippet of an XML Job Info string referring to files to be submitted to Express Web Services directly.

```xml
<JobInfo>
  <JobSettings JobID="" UserJobID="" JobPriority="3" AdlibServer="" />
  <JobFileMsg Count="" />
  <JobFileList>
    <JobFile Folder="\\myserver\fileshare" FileName="MyFile.doc" />
  </JobFileList>
</JobInfo>
```

Here is a snippet of the XML Job Ticket that will be used in conjunction with the above XML Job Info String:

```xml
<JOB:DOCINPUTS>
  <JOB:DOCINPUT FILENAME="MyFile.doc" FOLDER="" />
</JOB:DOCINPUTS>
```

**Submitting Files using AddJobFolder**

Express Web Services implementations where the files that are to be processed are not available on a file share that can be accessed by either Express Manager or Express, the AddJobFolder method of the Express Web Service Adaptor will create a temporary JobFolder that can be used to transfer files. The steps involved in using this method of file exchange are as follows:

1. Call the AddJobFolder() method of the Express Web Service Adaptor. A unique JobFolder will be created with an Input and Output sub-folder. This method will return an XML string containing both an Input JobFolder and an Output JobFolder.

2. Transfer files to the Input JobFolder using HTTP or FTP.

3. Construct an XML Job Ticket that refers to the Job Files in the Input JobFolder.

4. Submit the Job to Express Web Service Adaptor using the AddJob() method.

5. The Express Web Service Adaptor transfers the Job to the Express Manager.

6. The Express Manager positions the Job in the JobInput Queue based on the Job Priority specified in the XML JobInfo.

7. The Express Manager locates the next available Express Server where the Job is to be processed.

8. The Job in the JobInput Queue is sent to the Express Connector on the next available Express Server.
9. The Express Connector extracts the Job Ticket and submits them to Express for processing.

10. Express accesses the files from the Input JobFolder for processing.

11. When the Job is complete, Express transfers the processed documents (PDF/Image/Text Files) to the Output JobFolder defined in the XML Job Ticket.

12. The Express Connector reports the Job Status (Success or Failure) back to the Express Manager. Job Status is reported back to the calling application via the Express Web Service Adaptor using the GetJobStatus() method.

14. The calling application must get the processed files from the Output JobFolder.

15. The Job can be removed from Express Web Services by calling the DeleteJob() method of the Express Web Service Adaptor once the job is completed. The DeleteJob() method will cleanup any files from the Input and Output JobFolders and remove the JobFolders.

### Submitting Files using Web Service Attachments

For Express Web Services implementations where the files that are to be processed are not available on a file share that can be accessed by either Express Manager or Express, the AddJobFolder method of the Express Web Service Adaptor will create a temporary JobFolder on the WSA Server that can be used to transfer files via DIME. The steps involved in using this method of file exchange are as follows:

1. Call the AddJobFolder() method of the Express Web Service Adaptor. A unique JobFolder will be created with an Input and Output sub-folder. This method will return an XML string containing both an Input JobFolder and an Output JobFolder.

2. Call the AddJobInputFilesAsAttachments() to Transfer files to the Input JobFolder using DIME.

3. Construct an XML Job Ticket that refers to the Job Files in the Input JobFolder.

4. Submit the Job to Express Web Service Adaptor using the AddJob() method.

5. The Express Web Service Adaptor transfers the Job to the Express Manager.

6. The Express Manager positions the Job in the JobInput Queue based on the Job Priority specified in the XML JobInfo.

7. The Express Manager locates the next available Express Server where the Job is to be processed.

8. The Job in the JobInput Queue is sent to the Express Connector on the next available Express Server.
9. The Express Connector extracts the Job Ticket and submits them to Express for processing.

10. Express accesses the files from the Input JobFolder for processing.

11. When the Job is complete, Express transfers the processed documents (PDF/Image/Text Files) to the destination specified (HTTP, FTP, Fileshare) JobFolder defined in the XML Job Ticket.

12. The Express Connector reports the Job Status (Success or Failure) back to the Express Manager. Job Status is reported back to the calling application via the Express Web Service Adaptor using the GetJobStatus() method.

14. The calling application must get the processed files from the destination (HTTP, FTP, Fileshare).

15. The Job can be removed from Express Web Services by calling the DeleteJob() method of the Express Web Service Adaptor once the job is completed. The DeleteJob() method will cleanup any files from the Input and Output JobFolders and remove the JobFolders.

16. Call the DeleteJobFolder() method to remove any existing space created by the AddJobFolder routine.

**Submitting Files using XML Stream**

For Express Web Services implementations where the files that are to be processed are not available on a file share that can be accessed by either Express Manager or Express, the AddJobFolder method of the Express Web Service Adaptor will create a temporary JobFolder on the WSA Server that can be used to transfer files via XML Stream. This method is not recommended for files sizes greater than 5 megabytes in size. The steps involved in using this method of file exchange are as follows:

1. Call the AddJobFolder() method of the Express Web Service Adaptor. A unique JobFolder will be created with an Input and Output sub-folder. This method will return an XML string containing both an Input JobFolder and an Output JobFolder.

2. Call the AddJobInputFilesAsString() to Transfer files to the Input JobFolder using XML Stream.

   *note: xml structure similar to the following:

   `<JobFiles><JobFile FileName="test.doc" Data="asdf34rndf" Length="2"/></JobFiles>`

   Data attribute is a base64 encoded string

3. Construct an XML Job Ticket that refers to the Job Files in the Input JobFolder.

4. Submit the Job to Express Web Service Adaptor using the AddJob() method.

5. The Express Web Service Adaptor transfers the Job to the Express Manager.

6. The Express Manager positions the Job in the JobInput Queue based on the Job Priority specified in the XML JobInfo.
7. The Express Manager locates the next available Express Server where the Job is to be processed.

8. The Job in the JobInput Queue is sent to the Express Connector on the next available Express Server.

9. The Express Connector extracts the Job Ticket and submits them to Express for processing.

10. Express accesses the files from the Input JobFolder for processing.

11. When the Job is complete, Express transfers the processed documents (PDF/Image/Text Files) to the destination specified (HTTP, FTP, Fileshare) JobFolder defined in the XML Job Ticket.

12. The Express Connector reports the Job Status (Success or Failure) back to the Express Manager. Job Status is reported back to the calling application via the Express Web Service Adaptor using the GetJobStatus() method.

14. The calling application must get the processed files from the destination (HTTP, FTP, Fileshare).

15. The Job can be removed from Express Web Services by calling the DeleteJob() method of the Express Web Service Adaptor once the job is completed. The DeleteJob() method will cleanup any files from the Input and Output JobFolders and remove the JobFolders.

16. Call the DeleteJobFolder() method to remove any existing space created by the AddJobFolder routine.

**Receiving Processed Files**

Files that are processed by Express Server are created and delivered according to the settings in the XML Job Ticket.

**UNC Path**

The following is a snippet of an XML Job Ticket that shows the creation of a PDF in a folder:

```xml
<JOB:DOCOUTPUTS>
  <JOB:DOCOUTPUT FILENAME="MyDocument.jpg.pdf" FOLDER="" DOCTYPE="PDF"/>
</JOB:DOCOUTPUTS>
```

If multiple folders are required, then a DESTINATION element must be used. The following is a snippet of an XML Job Ticket that shows the creation of a PDF in multiple folders:

```xml
<JOB:DOCOUTPUTS>
  <JOB:DOCOUTPUT FILENAME="MyDocument.jpg.pdf" FOLDER="" DOCTYPE="PDF"/>
</JOB:DOCOUTPUTS>
<JOB:DESTINATION>
  <JOB:DESTFOLDERS>
    <JOB:DESTFOLDER FOLDER= "\\MyServer\Folder1" />
  </JOB:DESTFOLDERS>
</JOB:DESTINATION>
```
URI

The following is a snippet of an XML Job Ticket file showing an HTTP URI:

```xml
<JOB:DESTINATION>
  <JOB:DESTURIS>
    <JOB:DESTURI PROTOCOL="HTTP" HOST="http://myserver" PORT="80"
      FOLDER="exponentJobs/820200414/Output/" REQUESTTIMEOUT="60" />
  </JOB:DESTURIS>
</JOB:DESTINATION>
```

**GetJobFilesAsAttachments()**

Use the GetJobFilesAsAttachments web service method to retrieve any files specified on the IIS server as a DIME attachment.

**GetJobFilesAsStream()**

Use the GetJobFilesAsStream web service method to retrieve any files specified on the WSA Server as an XML stream.
The following is a snippet of an XML Job Ticket file showing an FTP URI:

```xml
<JOB:DESTINATION>
    <JOB:DESTURIS>
        <JOB:DESTURI PROTOCOL="FTP" HOST="www.myftpserver.com" PORT="21"
            FOLDER="MyDocuments" USERNAME="Adlib" PASSWORD="publ1sh" />
    </JOB:DESTURIS>
</JOB:DESTINATION>
```
Appendix A – System Requirements

Express Web Service Adaptor

<table>
<thead>
<tr>
<th>Hardware Requirements</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor Type</td>
<td>Minimum: Pentium 4 compatible processor</td>
</tr>
<tr>
<td>Processor Speed</td>
<td>Minimum: 1 GHz</td>
</tr>
</tbody>
</table>
| Memory (RAM)         | Minimum: 512 MB  
|                      | Recommended: 2 GB or higher |
| Hard Disk Space      | Minimum: 10 MB available |

<table>
<thead>
<tr>
<th>Software Requirements</th>
<th></th>
</tr>
</thead>
</table>
| Operating System      | Microsoft Windows Server 2003 32-bit  
|                       | Microsoft Windows Server 2003 64-bit |
| .NET Framework        | .NET Framework 2.0 (with latest Service Pack) or greater  
| IIS                   | Internet Information Services (IIS) must be installed |
| ASP.NET Service       | Must be installed and enabled for Windows Server 2003 |
| WebDAV                | Must be allowed to process PUT commands (default is prohibited) |
| WSE 3.0               | Web Service Enhancements 3.0 |

Express Manager

<table>
<thead>
<tr>
<th>Hardware Requirements</th>
<th></th>
</tr>
</thead>
</table>
| Processor Type       | Minimum: Pentium 4 compatible processor  
|                      | Recommended: Dual or Quad processors |
| Processor Speed      | Minimum: 1 GHz  
|                      | Recommended: 2.8 GHz or higher |
| Memory (RAM)         | Minimum: 512 MB  
|                      | Recommended: 2 GB or higher |
| Hard Disk Space      | Minimum: 10 MB available |

<table>
<thead>
<tr>
<th>Software Requirements</th>
<th></th>
</tr>
</thead>
</table>
| Operating System      | Microsoft Windows Server 2003 32-bit  
|                       | Microsoft Windows Server 2003 64-bit |
| .NET Framework        | .NET Framework 2.0 (with latest Service Pack) or greater  
| Message Queuing       | Message Queuing (MSMQ) version 2.0 or greater |

**Note:**  
It is recommended that Express Manager and Express Connector(s) be installed on separate systems.
<table>
<thead>
<tr>
<th><strong>Express Connector</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hardware Requirements</strong></td>
</tr>
<tr>
<td>Processor Type</td>
</tr>
<tr>
<td>Processor Speed</td>
</tr>
<tr>
<td>Memory (RAM)</td>
</tr>
<tr>
<td>Hard Disk Space</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Express Connector</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operating System</strong></td>
</tr>
<tr>
<td>Microsoft Windows Server 2003 32-bit</td>
</tr>
<tr>
<td>Microsoft Windows Server 2003 64-bit</td>
</tr>
<tr>
<td><strong>Message Queuing (MSMQ)</strong></td>
</tr>
<tr>
<td><strong>Express Server</strong></td>
</tr>
</tbody>
</table>
Appendix B – Software Installation

Express Web Service Adaptor

To install Express Web Service Adaptor, you should be logged on with Administrator’s rights and install from the local machine. Do not install from Terminal Services.

Refer to Appendix A – System Requirements to review the list of required software prior to installing Express Web Service Adaptor Manager.

To install the Express Web Service Adaptor please follow these steps:

1. Install the available Operating System Service Packs
   - For Windows 2000 install SP4 or greater
     http://www.microsoft.com/windows2000/
   - For Windows XP install SP1 or greater
     http://www.microsoft.com/windowsxp/downloads

2. Install the .NET runtime environment Version 2.0
   http://msdn.microsoft.com/netframework/downloads/

3. Install Internet Information Services (IIS)
   Refer to Appendix D – Installing IIS for detailed instructions.

4. Install Web Service Enhancements 2.0 Service Pack 3 (WSE) when using Legacy (DIME), or Web Service Enhancements 3.0 (WSE) when using the default WSA (recommended).

5. Run ‘ExpressWebServices.exe’ and follow the instructions on the screen.

6. Note that if installing Express Web Service Adaptor on a 64-bit version of Windows 2003 Server, the World Wide Web Publishing Service does not support running 32-bit and 64-bit worker processes concurrently on the same server. Installing Express Web Service Adaptor will enable running 32-bit processes, thus disabling any existing 64-bit processes. For more information on running 32-bit processes on a 64-bit OS, see the following:


7. Note that TCP Port 9932 (by default) is used by the Express Web Service Adaptor to communicate with the Express Manager. For Windows XP with Service Pack 2 installed you must configure your Windows Firewall to open this port.

   Follow these steps:
   
   1. Open the Control Panel from the Windows Start Menu.
   2. Select Network and Internet Connections.
4. Click on the **Exceptions** tab.
5. Click on the **Add Port** button.
   a. Enter Name: EXPONENT
   b. Enter Port Number 9932
   c. Select TCP
6. Click OK
7. Click OK to Save Changes

8. Note that WEBDAV is used by the Express Web Service Adaptor. When running

9. Failover Managers may be added to the list in the Web Service Adaptor. This will allow the Web Service Adaptor to automatically move to the next Manager if the first one is offline. Edit the “ExponentWSASettings.xml” file (by default, located at c:\inetpub\wwwroot\ExponentWSA), and add computers with Managers to the list:
   
   `<exponentManager ServerName="MachineName1" RemotingPort="9932" />`
   `<exponentManager ServerName="MachineName2" RemotingPort="9932" />`
Express Manager

To install Express Manager, you must be logged on with Administrator’s rights and install from the local machine. **The Express Manager service needs to logon as a local administrator.** Do not install from Terminal Services.

Refer to **Appendix A – System Requirements** to review the list of required software prior to installing Express Manager.

1. Install the available Operating System Service Packs
   - For Windows 2000 install SP4 or greater
   - For Windows XP install SP1 or greater

2. Install the .NET runtime environment Version 2.0

3. Install Microsoft Message Queue (MSMQ)
   Refer to **Appendix E – Installing MSMQ** for detailed instructions.

4. Run ‘ExpressWebServices.exe’ and follow the instructions on the screen.

5. Select Express Manager from the Programs menu in Windows to start Express Manager.

6. Select Express Manager from the Programs menu in Windows to start Express Manager.

7. Note that HTTP Port 9932 is used by the Express Web Service Adaptor to communicate with the Express Manager.
Express Connector

Repeat the following steps for each individual machine that an Express Connector is installed on:

1. Install the available Operating System Service Packs
   - For Windows 2000 install SP4 or greater
     http://www.microsoft.com/windows2000/
   - For Windows XP install SP1 or greater
     http://www.microsoft.com/windowsxp/downloads

2. Install the .NET runtime environment Version 2.0
   http://msdn.microsoft.com/netframework/downloads/

3. Install Express Server 3.2.28 or greater.
   Refer to Express Server Configuration for details about how to configure Express to function properly with Express Connector.

4. Install Microsoft Message Queue (MSMQ)
   Refer to Appendix E – Installing MSMQ for detailed instructions.

5. Run "ExpressWebServices.exe" and follow instructions on the screen.

6. The Express Connector service needs to logon as a local administrator.

7. Select Express Connector from the Programs menu in Windows to start Express Connector.
Express Server Configuration

The following setting must be configured in Express Server so that it can operate correctly with Express Connector:

- **Include Subfolders**
  - must be enabled

- **Input File Handling**
  - must be set to ‘Move File to Output Folder’

- **Enable Log Settings**
  - must be set to ‘enabled’
  - **Log Settings screen**
    - Process Log must be enabled
    - Document Log must be enabled

- **Process Upon Folder Change**
  - must be enabled
  - or
    - **Check Input Folder Every**
      - must be set to 10 Seconds or less (1 Second is recommended)

- **Only Process XML/DPI Job Tickets**
  - must be enabled

- **Job Ticket Handling**
  - must be set to ‘Move File to Output Folder’
Appendix C – Version History

Express Web Service Adaptor

Version 4.7.0
Resolved issue with legacy (WSE 2.0) WSA installs

Version 4.5.0
Added support for Windows Server 2003 64-bit

Version 4.1.0
Added 'GetJobIDs' call to WSA which retrieves a list of messages in one or more queues

Version 4.0.0
Branding and name change for packaging and integration with Express Server
Exponent name changed to Express Web Services.
Exponent Web Service Adaptor name changed to Express Web Service Adaptor

Version 2.5.2
Version increment only

Version 2.5.1
Root folder could not be specified in Silent Install CFG file

Version 2.5.0
MTOM (Message Transmission Optimization Mechanism)
  o Replaces DIME (although DIME is still available as a legacy option)
Express Manager Failover
  o Allow for backup Managers to be in the Express WSA list to act as a failover/redundancy architecture at the Manager level
Enhanced Logging
  o Logging is now more detailed, flexible and efficient (efficient memory usage, etc.)
.NET Framework 2.0 Technology (vs. 1.1)
Enhanced Job Load Balancing routine in Express Manager
  o Allow for a more evenly and logically distribution of jobs across Connectors/Express
Exponent Branding Change
Other minor bug fixes and enhancements

Version 2.0.0
Only works with Express 3.2.28 and above

Version 1.6.2
Update GetJobErrorDetail to return xml format

Version 1.6.1
DeleteJobFolder will delete files that are read only now
Version 1.5.0
Additional Web Service Methods
GetConnectorStatusList()

Version 1.3.0
Ability to stream files into a scratch location on the webserver (DIME attachments)
Additional Web Service Methods
GetJobFilesAsAttachment()
GetJobFilesAsStream()
GetJobFileList()
DeleteJobFolder()
AddJobFilesAsAttachments()
AddJobFilesAsStream()
GetJobPosition()

Version 1.2.0
Initial Release
Express Manager

Version 4.7.0
Various performance and stability improvements
No longer sends unlimited emails upon connector offline
Stylesheet reference issue resolved in Error xmls
Amalgamated job log files

Version 4.5.0
User Job ID now returned in E-mail Notification from Manager
Added support for Windows Server 2003 64-bit

Version 4.0.0
Branding and name change for packaging and integration with Express Server.
Exponent name changed to Express Web Services.
Exponent Web Service Adaptor name changed to Express Web Service Adaptor.

Version 2.5.2
When using JobInfo.xml’s JobFileList/JobFile process, the job status would stay in SystemHold endlessly without every failing (when a job file does not exist or could not be read). New settings put in place to give a timeout and retry count for SystemHold jobs (JobSystemHoldRetryScanRate & JobSystemHoldRetryLimit).

Version 2.5.1
Silent Install created default folders when specifying a different location
Silent Install starts service even though CFG file explicitly states not to start service
Reading and saving settings file was not thread-safe

Version 2.5.0
.NET Remoting communication between Manager and Connector
  - Faster, more efficient response times when sending and receiving job and server statuses, etc. (any communication between Manager/Connector)
  - Eliminates bloating of the MSMQ service with messages (old way of communicating statuses)
Acceptance of fully-qualified names in the Express Manager/Connector List
  - Can now connect Manager/Connector across domains by accepting fully qualified names of computers
Enhanced Logging
  - Logging is now more detailed, flexible and efficient (efficient memory usage, etc.)
.NET Framework 2.0 Technology (vs. 1.1)
Enhanced Job Load Balancing routine in Express Manager
  - Allow for a more evenly and logically distribution of jobs across Connectors/Express
Exponent Branding Change
Other minor bug fixes and enhancements

Version 2.0.0
Only works with Express 3.2.28 and above
Version 1.6.3
Updated Settings file to reflect optimized scan rates
Fixed issue with email feature
Trigger events
Fixed Logging retention issue

Version 1.6.2
Express Manager using wrong settings
Submit a problem incomplete xml

Version 1.6.1
Manager Silent Install added

Version 1.6.0
Express Manager Install - not registering mscomctl2.ocx
Express Manager Logging Routines
XML Settings File updated
Job Scheduling added

Version 1.5.0
Load Balancing improved
Scanning Intervals changed from seconds to milliseconds

Version 1.3.0
Server groups
Direct jobs to specific connectors
Email administrator when jobs fail and when connector’s go offline

Version 1.2.0
Initial Release
Express Connector

Version 4.5.0
Various performance and stability fixes
Resolved XSL reference issue in Error XMLs

Version 4.5.0
Licensing Removed
Added support for Windows Server 2003 64-bit
Auto-Configure Express configuration now chooses 'Process upon folder change'

Version 4.0.0
Branding and name change for packaging and integration with Express Server
Exponent name changed to Express Web Services.
Exponent Web Service Adaptor name changed to Express Web Service Adaptor

Version 2.5.2
When using JobInfo.xml's JobFileList/JobFile to send files through Exponent, the job would never get processed
When a JobInfo.xml priority is left blank, the job would never get processed (nor would it ever fail)

Version 2.5.1
Express Connector not properly assigning Express Default Configuration on install
Silent Install created default folders when specifying a different location
Silent Install starts service even though CFG file explicitly states not to start service
Reading and saving settings file was not thread-safe

Version 2.5.0
.NET Remoting communication between Manager and Connector
  o Faster, more efficient response times when sending and receiving job and server statuses, etc. (any communication between Manager/Connector)
  o Eliminates bloating of the MSMQ service with messages (old way of communicating statuses)
Acceptance of fully-qualified names in the Express Manager/Connector List
  o Can now connect Manager/Connector across domains by accepting fully qualified names of computers
Enhanced Logging
  o Logging is now more detailed, flexible and efficient (efficient memory usage, etc.)
.NET Framework 2.0 Technology (vs. 1.1)
Enhanced Job Load Balancing routine in Express Manager
  o Allow for a more evenly and logically distribution of jobs across Connectors/Express
Exponent Branding Change
Other minor bug fixes and enhancements

Version 2.0.0
Only works with Express 3.2.28 and above

Version 1.6.5
Only works with Express 3.2.22 and above
HTML reserved characters (ie. %20) bug fixed when uri download checked
Connector - Express Added XML Document Log attribute

**Version 1.6.4**
GetJobStatus bug fixed

**Version 1.6.3**
Update Unicode Issues
Only works with Express 3.2.8 and above

**Version 1.6.2**
Updated AdlibJobTicket.dll
Submit a Problem fixed

**Version 1.6.1**
Updated AdlibJobTicket.dll
Connector no longer updates actual job ticket using jobticketbuilder.dll
Connector appends a jobticket template before the </JOBS> element and express applies the template to the jobticket
Logging routines fixed

**Version 1.6.0**
Updated AdlibJobTicket.dll

**Version 1.5.0**
Multi threading with Copy URI option
Scanning Intervals changed from seconds to milliseconds
Download files from secure URI location(https valid and non valid certificates)

**Version 1.3.0**
Download files from URI location (http, ftp and network)

**Version 1.2.0**
Initial Release
Appendix D – Installing IIS

IIS is required for the Express Web Service Adaptor

1. Open the Control Panel from the Windows Start Menu.
2. Double-click the Add/Remove Programs icon.
3. Click on Add/Remove Windows Components.
4. Select Internet Information Services (IIS).

Click Next and continue to follow the installation instructions on your screen.
Appendix E – Installing MSMQ

MSMQ is required for Express Manager, and Express Connector

To install MSMQ, navigate to Add/Remove Programs. Click Add/Remove Windows Components *note: on left hand side

Windows 2000 Server
1. Select Message Queuing Services → Next
2. Select Message Queuing Server → Next
3. Select access mode to active directory → Next
4. Select Message Queuing will not access a directory server → Next

Windows XP
1. Select Message Queuing Services → Details
2. Select only the Common option → OK
3. Next → Finish

Windows 2003
1. Select Application Server → Details
2. Select Message Queuing → Details
3. Select common and deselect everything else → Ok → Ok → Next

Windows 2000
1. Select Message Queuing Server → Next
2. Select Manually select access mode to active directory → Next
3. Select Message Queuing will not access a directory server → Next
Appendix F – Troubleshooting

1. If a user keeps receiving job failed as a message, please ensure that all directories that are being used to process a job are shared and have adequate permissions to read/write/execute. Also ensure that the path being specified is an appropriate UNC path.

2. If you are having problems uploading to a Windows 2003 IIS location, ensure that WEBDAV is allowed.
   a. Open the IIS manager, and select Web Service Extensions Folder
   b. Webdav should read allowed, not prohibited

3. Remote Desktop Implementations
   a. Express installed as service and runs under local administrator user
   b. Express Web Services products run as the same user that Express is running under

4. AddJob returns
   a. "Not a valid Adlib Express job ticket"
   This means your job ticket does not have the appropriate headers in order to be processed by Express Server. It is recommended to have the following headers:

   ```xml
   <?xml version="1.0" encoding="ISO-8859-1"?>
   <?AdlibExpress applanguage="USA" appversion='2.9' dtdversion='1.0' ?>
   <IDOCTYPE JOBS SYSTEM "C:\Adlib Express\DTD\AdlibExpress.dtd">
   ...
   </JOBS>
   ...
   </AdlibExpress>
   ...
   </xml>
   ```

5. The connector/manager will not process my jobs. The job remains pending for several hours. This could be due to insufficient disk space. Express Manager/Connector have a defined limit of 1 Gigabyte minimum in order to process a job. This is definable through the settings for each product. To resolve this, clear more than 1 Gigabyte of space on the system, or change the setting:

   - Edit the ExponentManagerSettings.xml and/or ExponentConnectorSettings.xml file (in their respective installation folders, which default to C:\Program Files\Adlib\Express Web Services\...)
   - Find the following attribute and decrease the value:
     - AllocatedDiskSpace="1000000000"

6. If the following is returned from invoking the AddJob web method, then your Manager is not running as a service.

   ```xml
   <ServerInfo>
   <Server Name="localhost" />
   <Error Description="Manager Offline" Message="No connection could be made because the target machine actively refused it" />
   </ServerInfo>
   ```
Appendix G – Express Web Services Problem Submission

The Submit Problem to Adlib screen simplifies the problem reporting process to Adlib by automating the assembly of information and packaging it for submission (see Figure 17 - Submit Problem to Adlib Screen). The information gathered includes contact information, problem description, problem files, Express settings and process logs as well as system information. We recommend that you attach all files associated with the problem including the following if applicable:

- Source Document
- Rendered Documents (PDF, TIFF and Text)
- XML Job Ticket Files/XML Job Info Tickets

This information is packaged into a single zip file that must be emailed to Adlib (support@adlibsoftware.com) for investigation. We recommend that you review the files included in the zip file prior to submission. Please be assured that Adlib shall hold your submitted information and documents in strict confidence.

Figure 17 - Submit Problem to Adlib Screen
Appendix H – Definitions

Adlib Server
An Adlib Server application which accepts Job Tickets and processes documents. Express and Adlib inSight are both Adlib Server applications.

DIME
Direct Internet Message Encapsulation. Web service adaptor (Legacy version) uses DIME and WS-Attachments to stream data. Web service adaptor is using Web Service Enhancements Version 2.0. This is replaced by MTOM (see MTOM).

Exponent
The old name for Express Web Services

Express Web Services
The suite of software applications which includes Express Manager, Express Adaptors and Express Connectors.

Express Adaptor
The software application which facilitates the communication of Jobs from disparate sources to the Express Manager.

Express Connector
The software application which receives JobMsgs from the Express Manager for processing. The Job is processed by Express or Adlib inSight on the same server.

Express Manager
The software application which accepts new Jobs from external entities and routes them according to job priority to the appropriate server.

Job
Consists of a JobMsg and optionally one or more JobFileMsg(s) which are related to each other by the JobID.

JobFile
A File which is referred to by a JobTicket that is to be sent to the Adlib Server via the Express Manager.

JobFileChunk
Job Files are compressed using zip compression and output to a multiple part zip file for sending via MSMQ Messaging. The part zip file is called a JobFileChunk.

JobFileMsg
An MSMQ Message which contains JobFileChunk.

JobID
A GUID derived from the MsgID when a message is placed in the JobRequest Queue. The JobID shall be stored in the CorrelationID property of the JobMsg and the JobFileMsg.

JobInfo
XML that stores and accumulates information about a job throughout the lifecycle of the Job. JobInfo stores information about how to process a Job as well as the status of the Job. The XML is stored in the Extension property of the MSMQ MsgHeader.
JobMsg
An MSMQ Message which contains a JobInfo XML and a JobTicket XML and is ready for processing.

JobRequestMsg
An MSMQ Message which contains a JobInfo XML and a JobTicketXML. The JobRequest is not a JobMsg until it has been prepared for processing and placed in the JobInput Queue.

JobTicket
An XML or Text based file that contains processing instructions to be passed to an Adlib Server.

MTOM
Message Transmission Optimization Mechanism, a method of efficiently sending binary data to and from web services. It uses XOP (XML-binary Optimized Packaging) to transmit binary data and is intended to replace both MIME and DIME attachments.

UserJobID
User defined JobID which is stored in the JobInfo XML.
Appendix I – Contact Adlib Software

Please contact Adlib Software if you have any questions or comments.

**Web:**  [www.Adlibsoftware.com](http://www.Adlibsoftware.com)

**E-mail:**  info@Adlibsoftware.com for General Inquiries
sales@Adlibsoftware.com for Sales Support
support@Adlibsoftware.com for Technical Support

**Phone:**  (905) 631-2875

**Fax:**  (905) 639-3540

**Mail:**  Adlib Software
215-3228 South Service Road
Burlington, Ontario L7N 3H8
Canada