

September 2008

Welcome to the September 2008 edition of IBM InfoSphere Information Server Developer's Notebook. This month we answer the question;

Can I retrieve and process Internet accessible files via methods other than Web services, and is it possible to parse and work with PDF files?

The short answers are; Yes, and Yes.

We retrieve and process Internet accessible files all of the time, generally with the intent of enriching data we already own. Using IBM InfoSphere Information Server, you can retrieve anything you view inside a Web browser using HTTP. The biggest concern with Web content is copyright infringement. In other words, its not a technical challenge one faces, its a legal challenge; are you legally allowed to take and process/re-process what you've retrieved?

PDF files are also easily managed with InfoSphere Information Server, as we demonstrate below.

Software versions

All of these solutions were *developed and tested* on (IBM) InfoSphere Information Server (IIS) version 8.1, using the Microsoft Windows XP/SP2 platform to support IIS client programs, and a RedHat Linux Advanced Server 4 (RHEL 4) FixPak U6 32 bit SMP server (Linux kernel version 2.6.9-67.EL-smp) to support the IIS server components.

IBM InfoSphere Information Server allows for a single, consistent, and accurate view of data across the full width of the corporate enterprise, be it relational or non-relational, staged or live data. As a reminder, the IBM InfoSphere Information Server product contains the following major components;

WebSphere Business Glossary Anywhere™, WebSphere Information Analyzer™, WebSphere Information Services Director™, WebSphere DataStage™, WebSphere QualityStage™, WebSphere Metadata Server and Metabridges™, WebSphere Metadata Workbench™, WebSphere Federation Server™, Classic Federation™, Event Publisher™, Replication Server™, Rational Data Architect™, DataMirror Transformation Server™, and others.

Obviously, IBM InfoSphere Information Server is a large and capable product, addressing many strategic needs across the enterprise, and supporting different roles and responsibilities.

21.1 Terms and core concepts

In this document, (IBM) InfoSphere Information Server Developer's Notebook (IISDN), we regularly discuss Web services, Service Oriented Architectures (SOA), and related topics. But what if a network or Web based resource you wish to access is not available via Web services, FTP, or a similar protocol? What if the resource (file) is only available via HTTP?

Figure21-1 displays a sample (IBM) InfoSphere Information Server (IIS) DataStage/QualityStage component Job, that reads a data file from a public Web site using HTTP. All you really need in Figure21-1 is the upper left most (blue) stage, the External Source stage. The External Source stage is enough to read the given file via HTTP. You can read text files, sound and image files, basically anything.

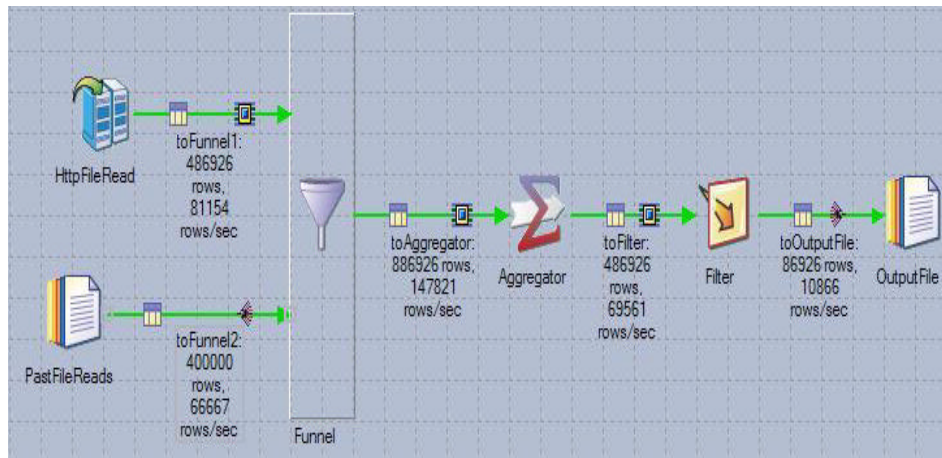


Figure 21-1 Sample HTTP File Read Job.

Comments related to the remainder of the DataStage/QualityStage Job in Figure21-1 include;

- As this DataStage/QualityStage Job existed in the real world, a U.S. government state agency appends data in the form of ASCII text to a given file, which is served through a state agency Web server. That file is not linked from any page, you must know the full (and private) URL address to that file.
- As an HTTP accessible file, we can only get what we get, which is, we can call to retrieve the full and complete file. The HTTP communication protocol, and/or the state agency, did not offer an incremental change data capture capability.

Since the file is appended to, in this case daily, we take the full file read from the External Source stage (named HttpFileRead above), and merge its output with the accumulation of all past file reads.

We use the Aggregator stage to count duplicate rows, and then use the Filter stage to remove all rows we have previously received and processed.

In our case we checked for duplicate (full width) rows, but we just as easily have checked for duplicate key (smaller width) values or the like.

- Again, all you really need to retrieve a Web based resource (file), is the External Source stage above.

Working with PDF files

As we often do, we cite Wikipedia.com for the definition of our sub-topic, in this case, PDF files-

Portable Document Format (PDF) is a file format created by Adobe Systems in 1993 for document exchange. PDF is used for representing two-dimensional documents in a manner independent of the application software, hardware, and operating system.

http://en.wikipedia.org/wiki/PDF_file

We'll also mention, PDF files are binary, not ASCII text.

(IBM) InfoSphere Information Server (IIS) is known to work with free form text, sometimes referred to as *unstructured data*, in the area of the QualityStage component. QualityStage can easily identify and cleanse information hidden inside free form text fields to find things like; apartment numbers offset from the mailing address proper, instructions or measurements detached from a parts record, and more.

As an instructional topic, and in the larger sense, we are discussing taking a PDF file and parsing it; taking its contents, standardizing it, and placing information into known and consistent data fields. That is a QualityStage topic in the classic sense. Further, we would likely create a QualityStage *custom rule set*, to understand and parse (standardize) our input PDF data source. We use the DataStage External Source stage to read the PDF file proper, and then use the QualityStage Standardize stage to parse that output.

You can use a QualityStage custom rule set to do this work, but you don't have to use a QualityStage custom rule set to do this work.

Note: A future edition of (IBM) InfoSphere Information Server Developer's Notebook (IISDN) will cover how to create a QualityStage custom rule set.

Today, we are going to parse the output of the source PDF file using a simple (Parallel) Transformer stage.

Using curl

As an acronym, curl stands for, *copy URL*. If you're not already familiar with the term, URL is itself an acronym, standing for, uniform resource locator. In our context, a URL is the fully qualified Web address of something we wish to view or retrieve.

In the physical sense, curl is also an operating system program, used to copy (files) over the Web. The curl program is pre-installed on most operating systems. Microsoft Windows is about the only operating system we encounter that does not have curl pre-installed.

If your operating does not include curl, curl may be retrieved here,

<http://curl.haxx.se/dlwiz/?type=bin>

By means of example, experiment with the following;

- Using a Web browser, open any Web site. By means of example, we opened <http://www.Ask.com>.
- View the HTML page source for the page you just retrieved.
In Firefox and from the Menu Bar, select, View -> Page Source.
In Internet Explorer, select, View -> Source.
- Any modern Web page will be using Cascading Style Sheets (CSS files). CSS files are ASCII text and smallish in size. In the HTML source code you just viewed, find any embedded CSS file reference. (Use an Edit -> Find, and look for ".css".)

While this is likely to change between we writing and you reading this document, our Ask.com home page includes a reference to a CSS file with the URL of,

<http://www.ask.com/dcss/a11/skinshp.r32645.css>

- From an operating system command prompt, enter the following,

```
curl http://www.ask.com/dcss/a11/skinshp.r32645.css
```

In response, you will receive the contents of that file. Easy squeezezy.

The curl program is very handy, handier still when invoked from the External Source stage of DataStage, as the output from curl can then be managed, cleansed, aggregated, whatever.

More comments related to curl;

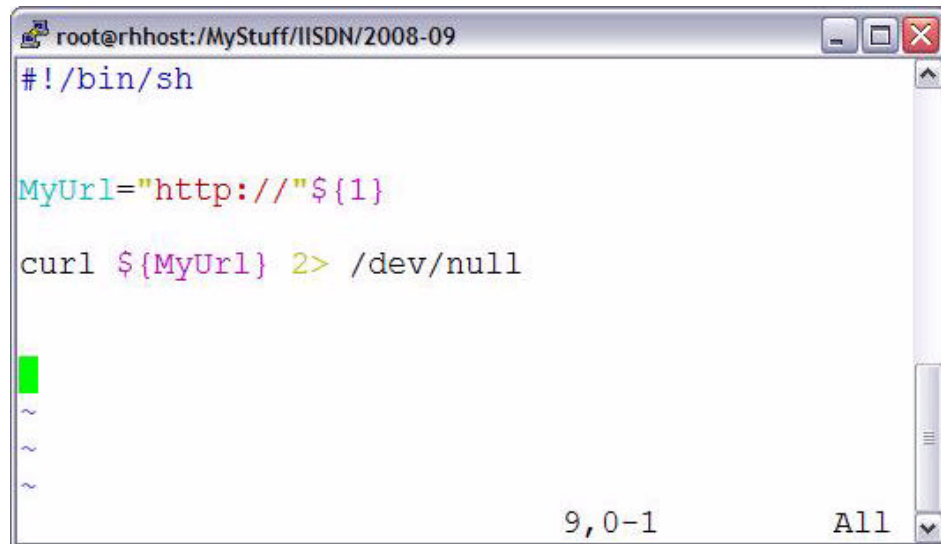
- Have you ever used a Web browser to open files on your hard disk? A URL can refer to remote or local URL's. Hence, curl can be used for remote or local URL's. Example,

```
curl http://www.SomeSite.com/SomeFile.csv
```

```
curl file://MyDirectory/MyFile.xls
```

- The curl program does do one thing we don't care for. If the file copy takes more than (n) seconds, curl offers a progress bar of sorts. That progress bar will cause grief for DataStage/QualityStage. To prevent this behavior, normally we put a wrapper around curl, similar to that as displayed in Figure 21-2.

With the runtime environment that accompanies IBM InfoSphere Information Server, you can use the wrapper exemplified below on Unix, Linux, Windows, whatever.



```
root@rhost: /MyStuff/IISDN/2008-09
#!/bin/sh

MyUrl="http://"${1}
curl ${MyUrl} 2> /dev/null

~
~
~

9,0-1 All
```

Figure 21-2 Wrapper for the curl program, suppresses progress bar.

Using pdftotext (Xpdf package)

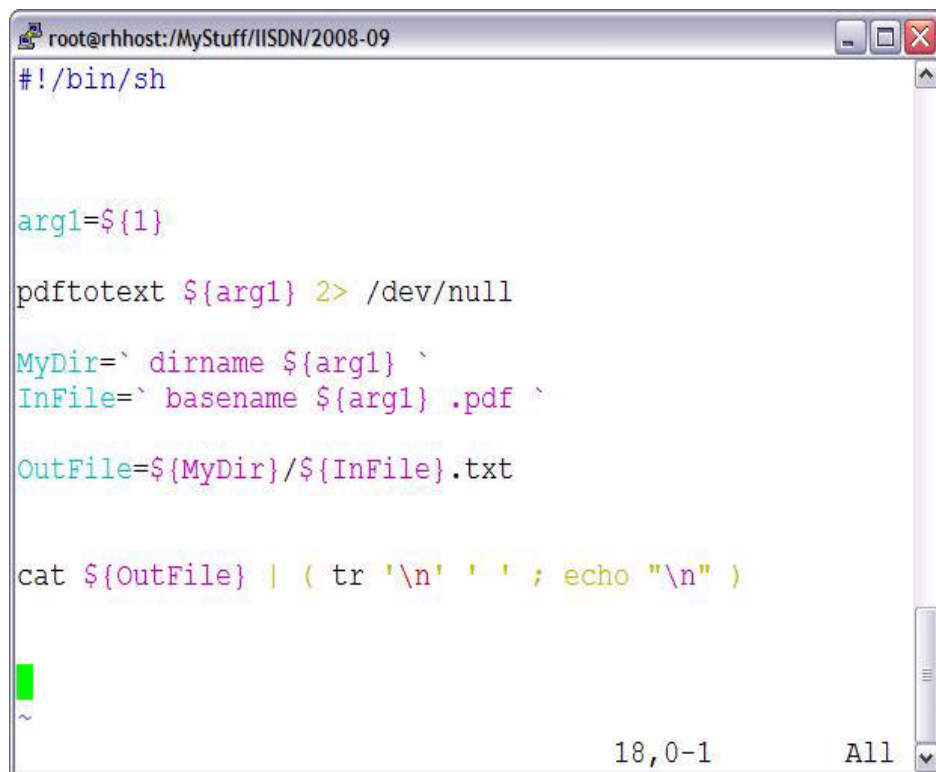
Similar to curl, (the Xpdf package) is an operating system set of utilities. Specifically, we like to use the pdftotext component to Xpdf. If your operating system does not include Xpdf, Xpdf may be retrieved here,

<http://www.foolabs.com/xpdf/download.html>

The pdftotext program component of the Xpdf package converts a PDF file to ASCII text. The Xpdf package also converts PDF to HTML (pdftohtml program), and more. Again we use the External Source stage to DataStage/QualityStage to invoke the utility of our choosing, in this case pdftotext, pdftohtml, whatever we need.

More comments related to pdftotext;

- Similar to our use of curl above, we wrapper pdftotext to fully control its output. Example as shown in, Figure21-3.



```

root@rhhost:/MyStuff/IISDN/2008-09
#!/bin/sh

arg1=${1}

pdftotext ${arg1} 2> /dev/null

MyDir=`dirname ${arg1}`
InFile=`basename ${arg1} .pdf`
OutFile=${MyDir}/${InFile}.txt

cat ${OutFile} | ( tr '\n' ' '; echo "\n" )

~
18,0-1 All

```

Figure 21-3 Wrapper for the pdftotext program, manges line feed characters.

- The pdftotext program outputs a filename based on the source PDF filename. Sincere we prefer to capture our output and send it to DataStage/QualityStage, we place a cat command inside our wrapper. The remainder of that wrapper constructs filenames, and manages line feed characters.

21.2 Sample Job using curl

In this section of this document, we create a simpler version of the DataStage/QualityStage Job displayed in Figure21-1. In order to complete this example as written, you will need the following;

- Internet access.
- The curl program needs to be present.
- A wrapper for the curl program, similar to that displayed in Figure21-2, needs to be present on the server tier for IBM InfoSphere Information Server.

Steps to complete include;

1. Launch the DataStage/QualityStage Designer program. Create a new Parallel Job, and save this Job with a given name.
2. From the Palette view, drag and drop the following stages;
 - a. From the Palette view, File drawer, 1 (count) External Source stage.
 - b. From the Palette view, File drawer, 1 (count) Sequential File stage.Rename and position the stages and links as displayed in Figure21-4.

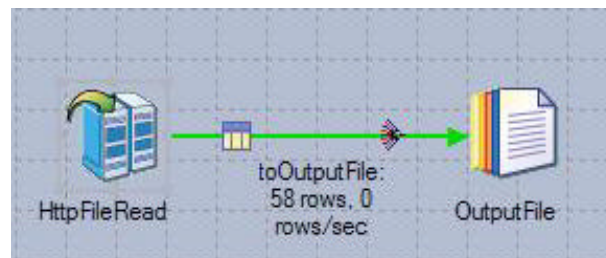


Figure 21-4 PJ03_ReadFromHttpSimple Job.

3. Set the following properties for the External Source stage (HttpFileRead above);
 - a. Output TAB -> Properties TAB -> Source Program.

We set ours equal to,

```
/MyStuff/IISDN/2008-09/HttpFileRead.sh  
www.ask.com/dcsc/a11/skinshp.r32645.css
```

That entry appears on one line, where the first (word) is the absolute file pathname to our curl program wrapper. And the second (word) is the variable part of the URL we wish to retrieve.

We got this URL from the example discussed above, a CSS file from the www.Ask.com Web site.

b. Figure 21-5 displays changes needed to the Output TAB -> Format TAB.

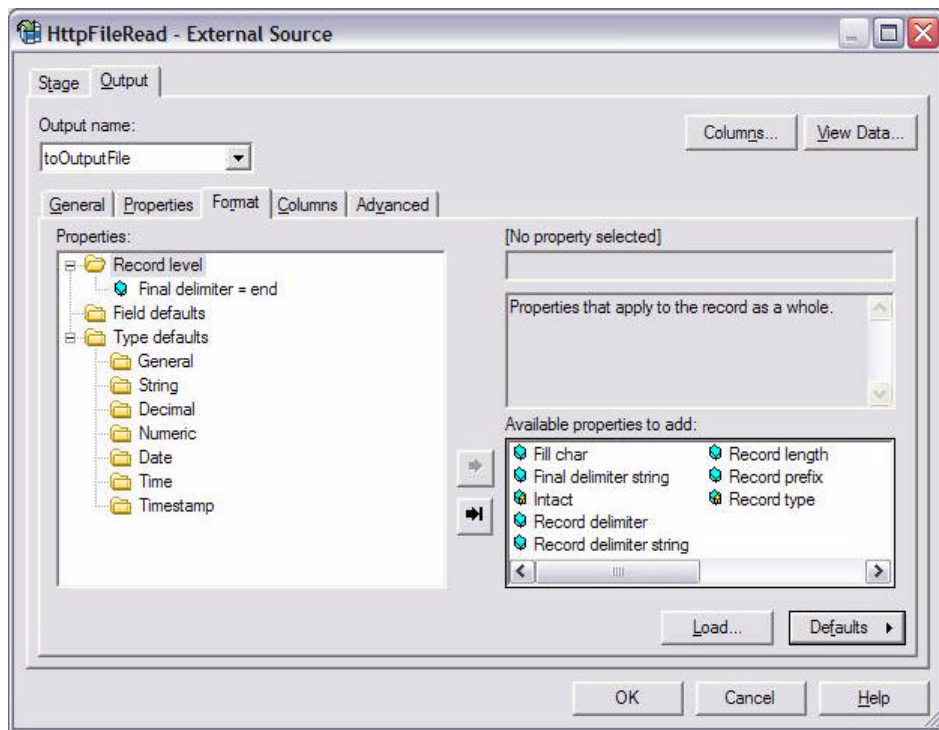


Figure 21-5 PJ03_ReadFromHttpSimple Job, Output TAB -> Format TAB.

c. And Figure21-6 displays changed needed to the Output TAB -> Columns TAB.

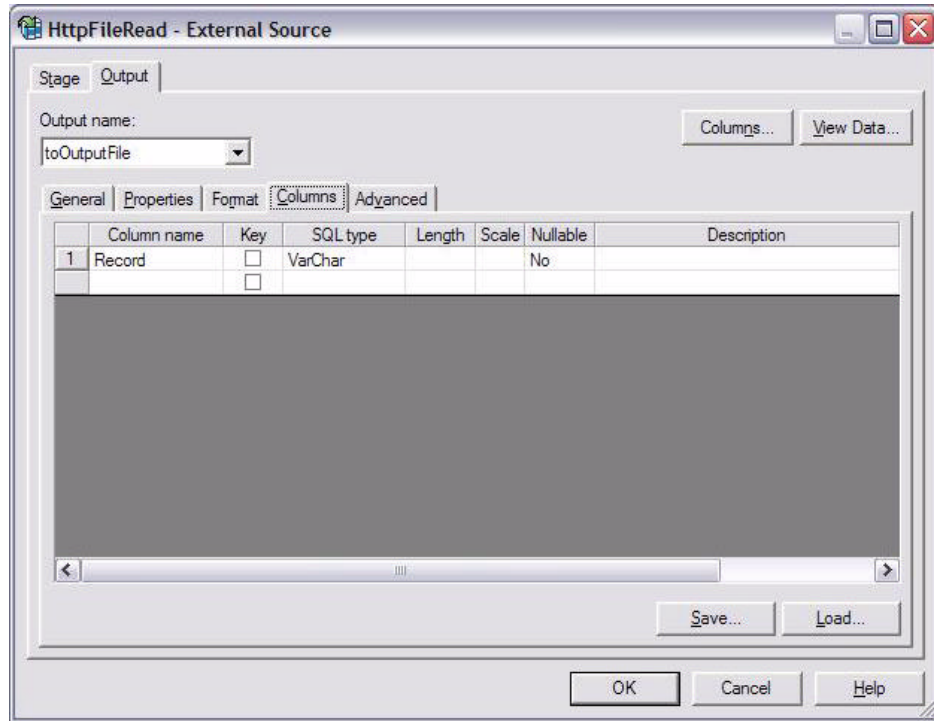


Figure 21-6 PJ03_ReadFromHttpSimple Job, Output TAB -> Columns TAB.

4. Set the Output File Name for the Sequential File stage, Save, Compile and Test.

A successful test looks like that as displayed in Figure21-4.

21.3 Sample Job using (Xpdf package) pdftotext

In this section of this document, we create a DataStage/QualityStage Job that reads a PDF file, parses it, and outputs select textual elements from that document. In order to complete this example as written, you will need the following;

- The Xpdf package needs to be present.
- A wrapper for the pdftotext program, similar to that displayed in Figure 21-3, needs to be present on the server tier for IBM InfoSphere Information Server.
- In order to follow these exact instructions, you need the same or similar PDF source file to the one we are using. We retrieved our file from

Amazon.com, a movie listing. We then used a print driver to output the (Web page) as a PDF file.

Our sample PDF source file appears in Figure21-7.

- From this sample PDF file we will dynamically extract the movie title, individual (listed) actor names, and the director's name.

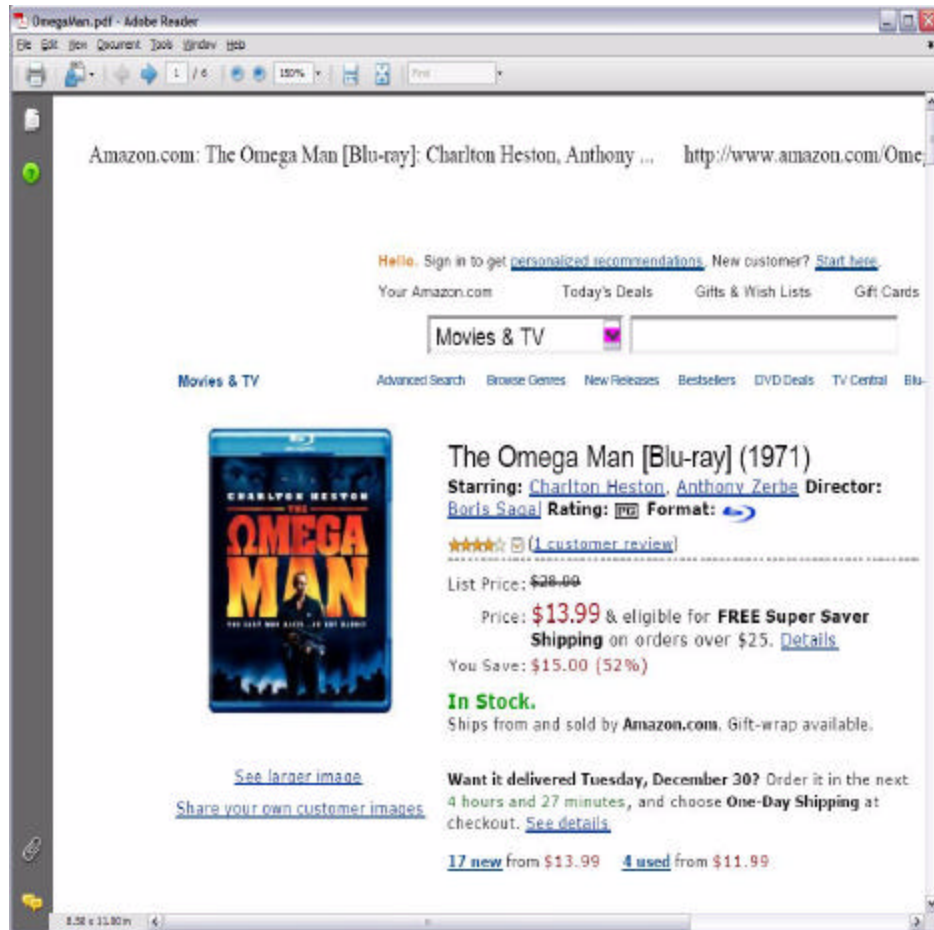


Figure 21-7 The Omega Man, coolest movie ever. Cooler than your favorite cool movie.

Steps to complete include;

5. Launch the DataStage/QualityStage Designer program. Create a new Parallel Job, and save this Job with a given name.
6. From the Palette view, drag and drop the following stages;
 - a. From the Palette view, File drawer, 1 (count) External Source stage.

- b. From the Palette view, File drawer, 1 (count) Sequential File stage.
 - c. From the Palette view, Processing drawer, 1 (count) Transformer stage.
- Rename and position the stages and links as displayed in Figure 21-8.

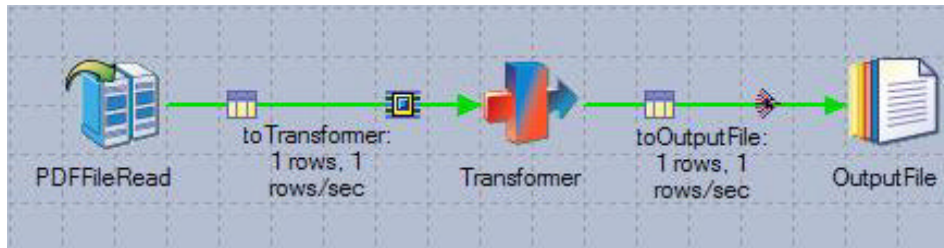


Figure 21-8 PJ11_PDFToData Job.

7. Set the following properties for the External Source stage (PDFFileRead above);

- a. Output TAB -> Properties TAB -> Source Program.

We set ours equal to,

```
/MyStuff/IISDN/2008-09/PDFFileRead.sh  
/MyStuff/IISDN/2008-09/OmegaMan.pdf
```

That entry appears on one line, where the first (word) is the absolute file pathname to our pdftotext program wrapper. And the second (word) is the variable file name we wish to convert.

- b. Figure 21-9 displays changes needed to the Output TAB -> Format TAB.
- c. And Figure 21-10 displays changes needed to the Output TAB -> Columns TAB.

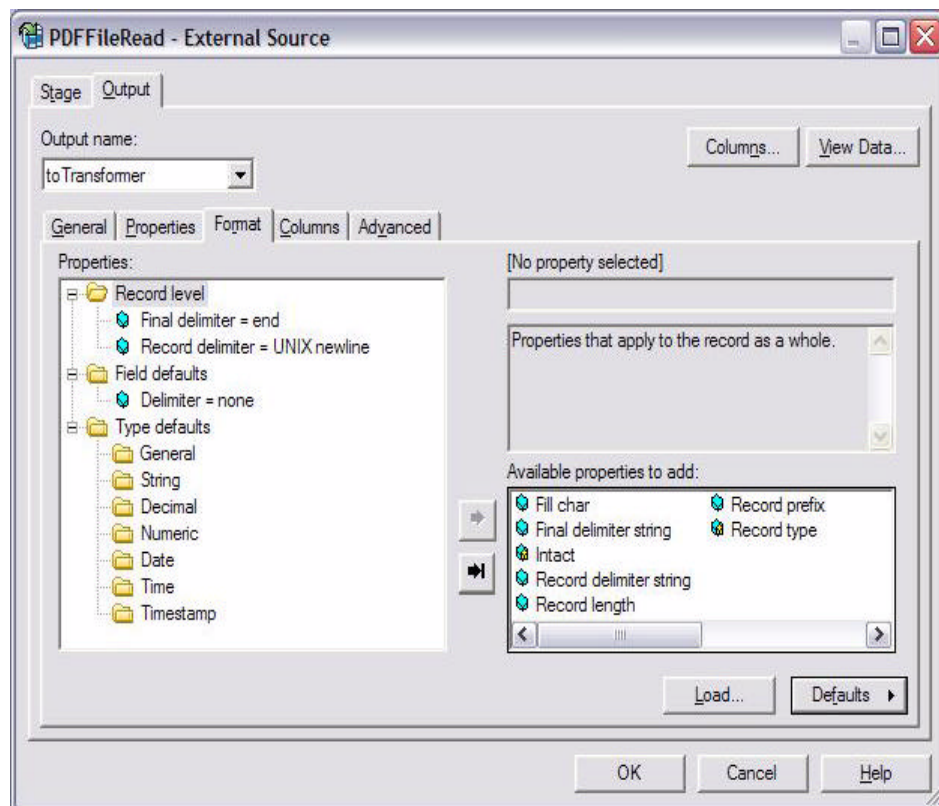


Figure 21-9 PJ11_PDFToData Job, Output TAB -> Format TAB.

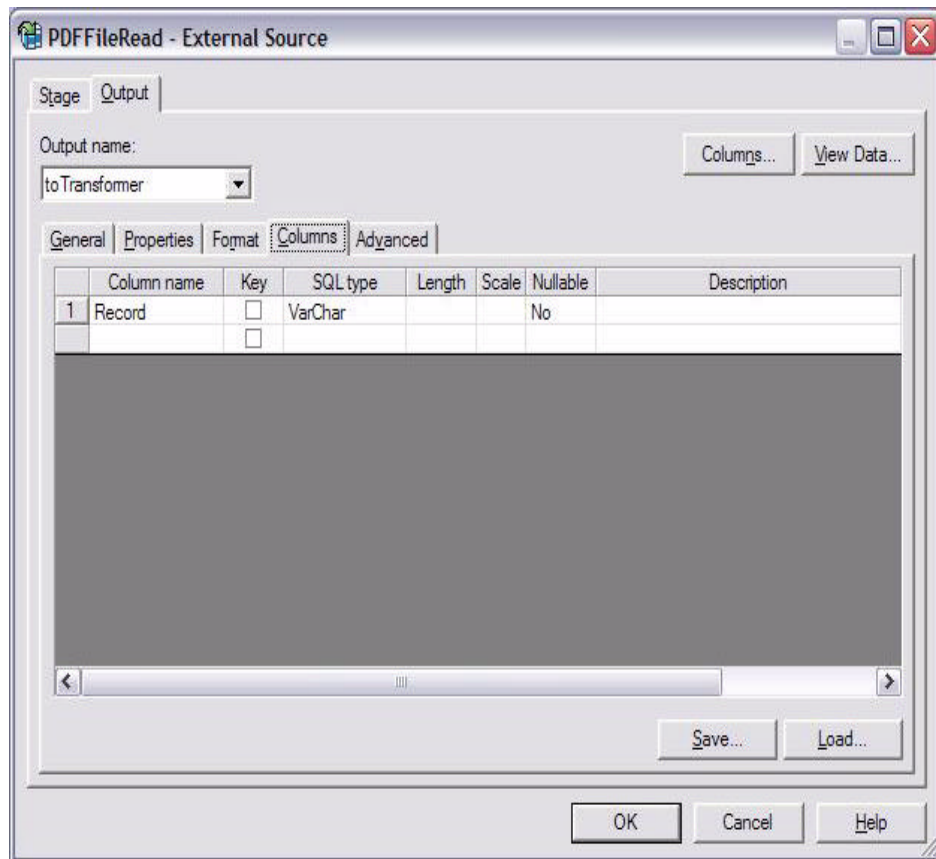


Figure 21-10 PJ11_PDFToData Job, Output TAB -> Columns TAB.

8. Set the following properties for the (Parallel) Transformer stage;
 - a. Create 25 (count) new Transformer variables.

Within the Transformer Properties dialog, Click the left most icon in the toolbar (shown with a red arrow in Figure21-11).

After you Click that toolbar icon, move to the Stage TAB -> Variables TAB.

Create these 25 variables with the variables names and other properties as displayed in Figure21-11.

You are done when your dialog box equals the display in Figure21-11.

Click OK when you are done.

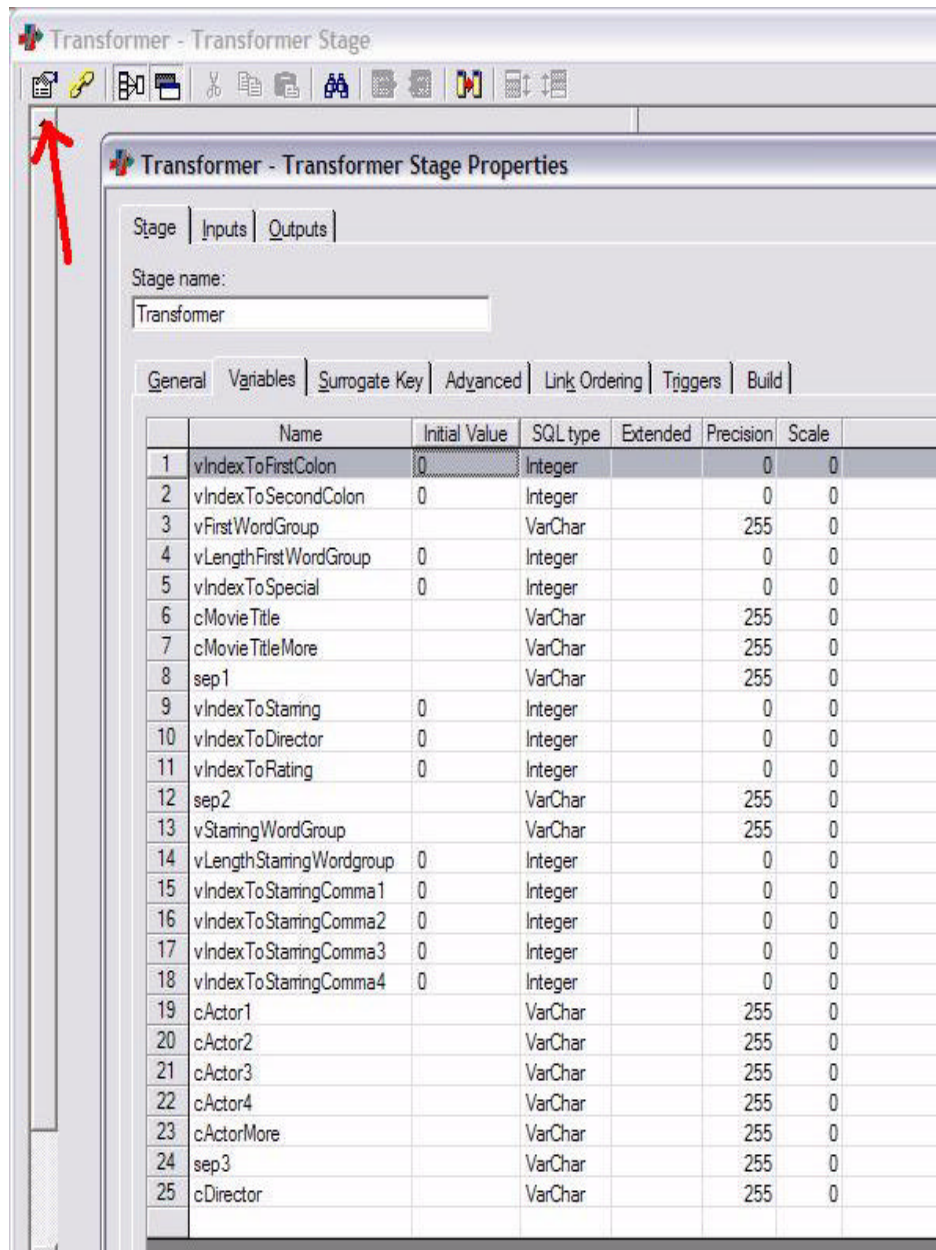


Figure 21-11 Creating (Parallel) Transformer variables.

- b. Back in the (Parallel) Transformer Properties dialog box proper, edit the Derivation Expression for each of the Transformer Variables you just created.

If you can not see the Stage Variables frame, as displayed in Figure 21-12, then you must first click the 'Show/Hide Transformer Variables' icon from the toolbar, shown with a red arrow.

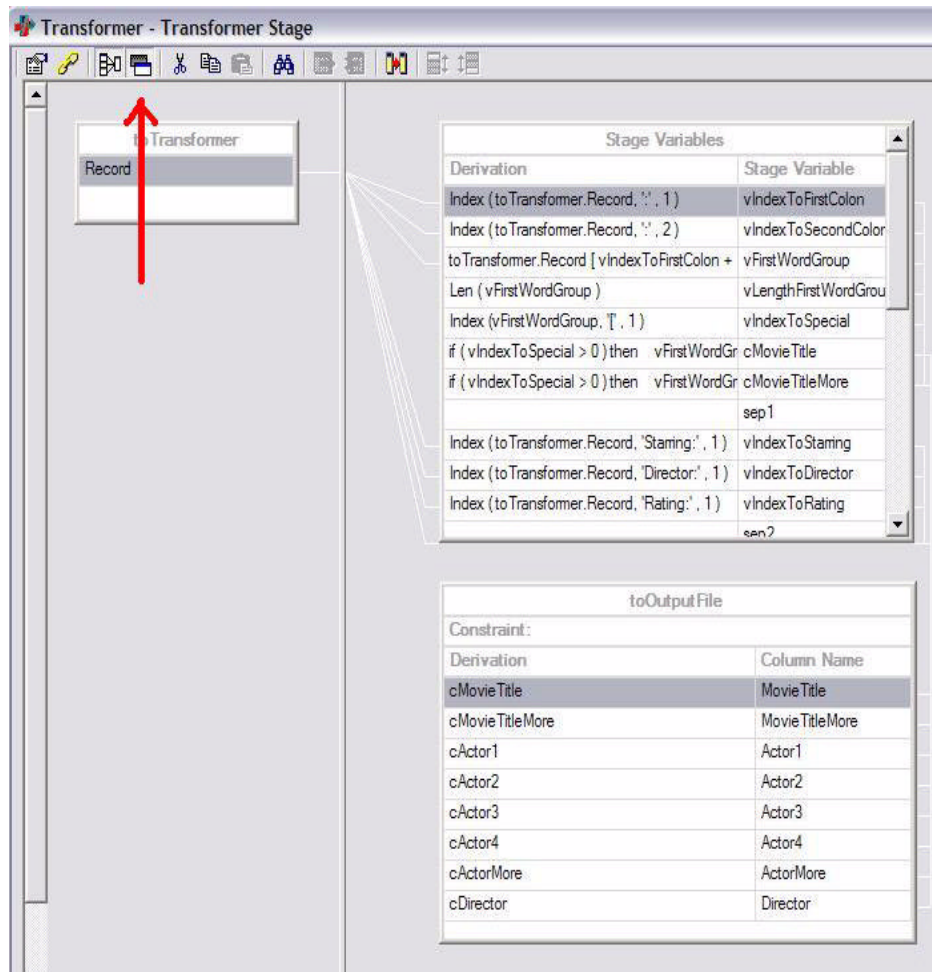


Figure 21-12 Setting (Parallel) Transformer Variable Derivation Expressions.

Each Transformer Variable, and its accompanying Derivation Expression, is listed below;

- i. vIndexToFirstColon

Index (toTransformer.Record, ':' , 1)

Code review for the above:

The first value we wish to extract from our outputted PDF text is the movie title. The movie title is delimited within the first two colons of this input stream. Here we get the position of the first colon.

ii. vIndexToSecondColon

Index (toTransformer.Record, ':' , 2)

Code review for the above:

Here we get the position of the second colon. (See above.)

iii. vFirstWordGroup

toTransformer.Record [vIndexToFirstColon + 1 ,
vIndexToSecondColon - vIndexToFirstColon - 1]

Code review for the above:

Within a variable we call FirstWordGroup, we extract the text between the first and second colon. This value will contain our movie title, and perhaps a bit more text.

iv. vLengthFirstWordGroup

Len (vFirstWordGroup)

Code review for the above:

Getting the length of the string above.

v. vIndexToSpecial

Index (vFirstWordGroup, '[', 1)

Code review for the above:

The movie title may variably contain a piece of text after it; for example, "My Movie [HD Format]".

Here we check for the presence of a "[" character.

vi. cMovieTitle

```
if ( vIndexToSpecial > 0 ) then
    vFirstWordGroup[ 1, vIndexToSpecial - 1 ]
else
    vFirstWordGroup
```

Code review for the above;

Here we extract the movie title. If there was a Special character, we extract one string, otherwise another.

vii. cMovieTitleMore

```
if ( vIndexToSpecial > 0 ) then
    vFirstWordGroup[ vIndexToSpecial + 1 , vLengthFirstWordGroup -
vIndexToSpecial - 1 ]
else
    , ,
```

Code review for the above:

If there was that Special movie title text, retrieve it here.

viii.sep1

We use the string "sep" to identify 'separators' within our variable list. This is just for style, to create line breaks between functional groups of variable expressions.

At this point we are done extracting the movie title.

ix. vIndexToStarring

Index (toTransformer.Record, 'Starring:' , 1)

Code review for the above:

The actor list is contained within the constants "Starring" and then "Director". The director is contained within the constants "Director", and then "Rating". Here we get positions for each of these strings, here and below.

x. vIndexToDirector

Index (toTransformer.Record, 'Director:' , 1)

xi. vIndexToRating

Index (toTransformer.Record, 'Rating:' , 1)

xii. sep2

xiii.vStarringWordGroup

toTransformer.Record [vIndexToStarring + 9 , vIndexToDirector - vIndexToStarring - 9]

Code review for the above:

Here we begin getting the actors list. Actor names are separated by commas. Will will extract up to 4 actor names, and then throw any overflow in the next column.

Here we just get the full list of actor names, unparsed.

```
xiv.vLengthStarringWordgroup  
    Len ( vStarringWordGroup )
```

Code review for the above:

Getting the length of the string above.

Note: We are really only using the same 3 or 4 (Parallel) Transformer Derivation Expression statements over and over. And for ease of reading at a later time (ease of maintenance later), we are being very verbose here; we could chop the volume of these statements in about half if we wished.

Also, we prefer to do all of our work (place all of our Derivation Expressions) alongside our variables whenever possible. We place them here versus placing them alongside the accompanying column.

The primary use case when you can not do this is when you are comparing a history of rows; the prior row to the current row.

```
xv.vIndexToStarringComma1  
    Index ( vStarringWordGroup, ',', 1 )
```

Code review for the above:

Here we get the location of the first comma in the actor list, second, third and so on, if present.

```
xvi.vIndexToStarringComma2  
    Index ( vStarringWordGroup, ',', 2 )
```

```
xvii.vIndexToStarringComma3  
    Index ( vStarringWordGroup, ',', 3 )
```

```
xviii.vIndexToStarringComma4
```

```
Index ( vStarringWordGroup, ',', 4)
```

```
xix.cActor1
```

```
if ( vIndexToStarringComma1 > 0 ) then
    vStarringWordGroup [ 1 , vIndexToStarringComma1 - 1 ]
else
    vStarringWordGroup
```

Code review for the above:

Here we get the first actor name, starting from position 1 in our string, to the location of the first comma, if present. If the first comma is not present, then we only have 1 actor, take the whole string.

```
xx. cActor2
```

```
if ( vIndexToStarringComma2 > 0 ) then
    vStarringWordGroup [ vIndexToStarringComma1 + 1 ,
vIndexToStarringComma2 - vIndexToStarringComma1 - 1 ]
else
    if ( vIndexToStarringComma1 > 0 ) then
        vStarringWordGroup [ vIndexToStarringComma1 + 1 ,
vLengthStarringWordgroup ]
    else
        ''
```

Code review for the above:

Here we get the second actor name. If there is a second comma, that means there is a third actor, read up until the second comma. If there is no second comma, there are only 2 actors, read until the end of the string.

```
xxi.cActor3
```

```
if ( vIndexToStarringComma3 > 0 ) then
```

```
vStarringWordGroup [ vIndexToStarringComma2 + 1 ,  
vIndexToStarringComma3 - vIndexToStarringComma2 - 1 ]  
else  
    if ( vIndexToStarringComma2 > 0 ) then  
        vStarringWordGroup [ vIndexToStarringComma2 + 1 ,  
vLengthStarringWordgroup ]  
    else  
        ''
```

Code review for the above:

Similar to second actor above; only the variable names have changed for third actor.

xxii.cActor4

```
if ( vIndexToStarringComma4 > 0 ) then  
    vStarringWordGroup [ vIndexToStarringComma3 + 1 ,  
vIndexToStarringComma4 - vIndexToStarringComma3 - 1 ]  
else  
    if ( vIndexToStarringComma3 > 0 ) then  
        vStarringWordGroup [ vIndexToStarringComma3 + 1 ,  
vLengthStarringWordgroup ]  
    else  
        ''
```

Code review for the above:

Similar to third actor above.

xxiii.cActorMore

```
if ( vIndexToStarringComma4 > 0 ) then  
    vStarringWordGroup [ vIndexToStarringComma4 + 1 ,  
vLengthStarringWordgroup - vIndexToStarringComma4 - 1 ]
```

```
else
```

```
    ''
```

Code review for the above:

Here we extract the remainder of the actor string, if there was a fourth comma (indicating the presence of 5 or more actors).

xxiv.sep3

xxv.cDirector

```
toTransformer.Record [ vIndexToDirector + 9 , vIndexToRating -  
vIndexToDirector - 9 ]
```

Code review for the above:

Getting the director name.

- c. Create 8 (count) new (Parallel) Transformer output columns, as displayed in Figure 21-13. Also, set the Column Derivations for these output columns as shown in Figure21-13.

You are done when your display equals that as shown in Figure 21-13.

Click OK when you are done.

toOutputFile						
Constraint:						
Derivation			Column Name			
cMovieTitle			MovieTitle			
cMovieTitleMore			MovieTitleMore			
cActor1			Actor1			
cActor2			Actor2			
cActor3			Actor3			
cActor4			Actor4			
cActorMore			ActorMore			
cDirector			Director			

toOutputFile						
	Column name	Key	SQL type	Length	Scale	Nullable
1	MovieTitle	<input type="checkbox"/>	VarChar			No
2	MovieTitleMore	<input type="checkbox"/>	VarChar			No
3	Actor1	<input type="checkbox"/>	VarChar			No
4	Actor2	<input type="checkbox"/>	VarChar			No
5	Actor3	<input type="checkbox"/>	VarChar			No
6	Actor4	<input type="checkbox"/>	VarChar			No
7	ActorMore	<input type="checkbox"/>	VarChar			No
8	Director	<input type="checkbox"/>	VarChar			No
		<input type="checkbox"/>				

Figure 21-13 Setting (Parallel) Transformer Output Columns.

- d. On the Sequential File stage, set the output file name, and any other properties you desire.
- e. Save the Job, Compile and Test.

A successful run appears equal to that as displayed in Figure21-14.

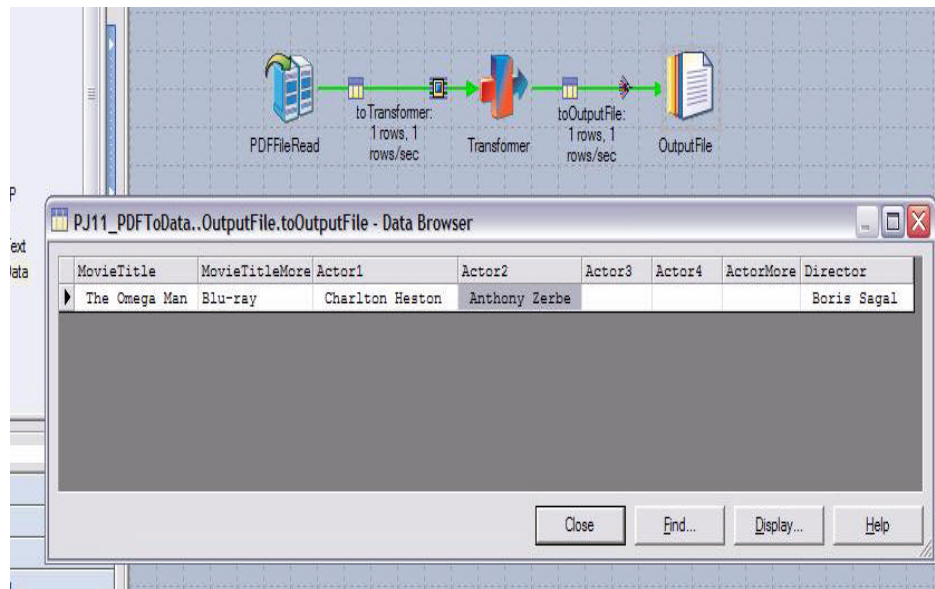


Figure 21-14 Successful Job results.

21.4 Summation

In this document, we reviewed or created:

We examined the (IBM) InfoSphere Information Server (IIS) capabilities in the area of retrieving Internet accessible files; specifically those not available via Web services, FTP, or similar means.

Also, we examined parsing of PDF files for the purpose of creating formatted output. While QualityStage Custom Rule Sets are probably the preferred technology here, with advanced word proximity, near spelling and Soundex matching, we chose to use a simple (Parallel) Transformer as that accomplished today's task quickly and easily.

Persons who help this month.

Steve Fazio, Andy Wilson, not Allen Spayth.

Additional resources:

The IBM WebSphere DataStage (InfoSphere Information Server), Web Services Pack Guide, version 8.1. (Filename, i46dewsv.pdf.)

(IBM) InfoSphere Information Server Developer's Notebook (IISDN), July 2007 edition; Web Services, Part 1.

(IBM) InfoSphere Information Server Developer's Notebook (IISDN), June 2007 edition; XML Stages.

Legal statements:

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating trademarks that were owned by IBM at the time this information was published. A complete and current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

Other company, product or service names may be trademarks or service marks of others.

Special attributions:

The listed trademarks of the following companies require marking and attribution:

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Microsoft trademark guidelines

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Intel trademark information

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

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

Other company, product, or service names may be trademarks or service marks of others.