

# IBM Datacap Accounts Payable Capture

Jan den Hartog

Tom Stuart



 Social

ECM





## Introduction to Accounts Payable Capture

The IBM® Datacap Accounts Payable Capture application is a *learning application*. Using a number of techniques, it has the ability to learn new instances of known documents when they are introduced into the system. The Accounts Payable Capture workflow can be used for many types of documents and applications, not just invoices.

This IBM Redpaper™ publication discusses the role that IBM Datacap Accounts Payable Capture plays in Accounts Payable (AP). You are introduced to the different jobs in the workflow and examine the task profiles, rulesets, rules, functions, and actions that make it work.

This paper guides you through the IBM Datacap Accounts Payable Capture application. The application is called a *foundation application* because it is used as a starting point for capturing complex machine-printed forms, such as invoices, that might contain line items.

Because Accounts Payable Capture is continually evolving with new technologies and techniques, your version of the product might differ slightly from what is shown in this paper. However, most of the techniques described in this paper are applicable to any version of the product. After you have gained an understanding of how these technologies interact, you will be able to apply them to other data capture scenarios needing similar capabilities.

## How Accounts Payable Capture works

IBM Datacap Accounts Payable Capture captures data from invoices and validates that data with business rules and with data stored in business applications. It then exports the document images, in PDF format, to the file system and the data to an XML file. In a production deployment, the document images would be exported to an image repository and the data to a business application.

In a typical application, a purchase order (PO) is entered into a Purchase Order system. After approval, the PO is sent to a vendor for fulfillment. The vendor ships the goods along with a shipping document.

The customer receives the goods and enters the shipping document into the business application as a Proof of Delivery (PoD). Next, the vendor sends an invoice to the Accounts Payable (AP) department. The AP department uses IBM Datacap Accounts Payable Capture to capture the invoice. Accounts Payable Capture interacts with the business application to obtain the most accurate data about the vendor and the items on the PO.

Once in the business application, a three-way match occurs. In this process, the PO line items, the PoD line items, and the invoice line items are compared. When these items match and the customer is confident that they have received the goods they ordered, the business application issues a check to the vendor.

Figure 1 illustrates this process in a simple diagram.

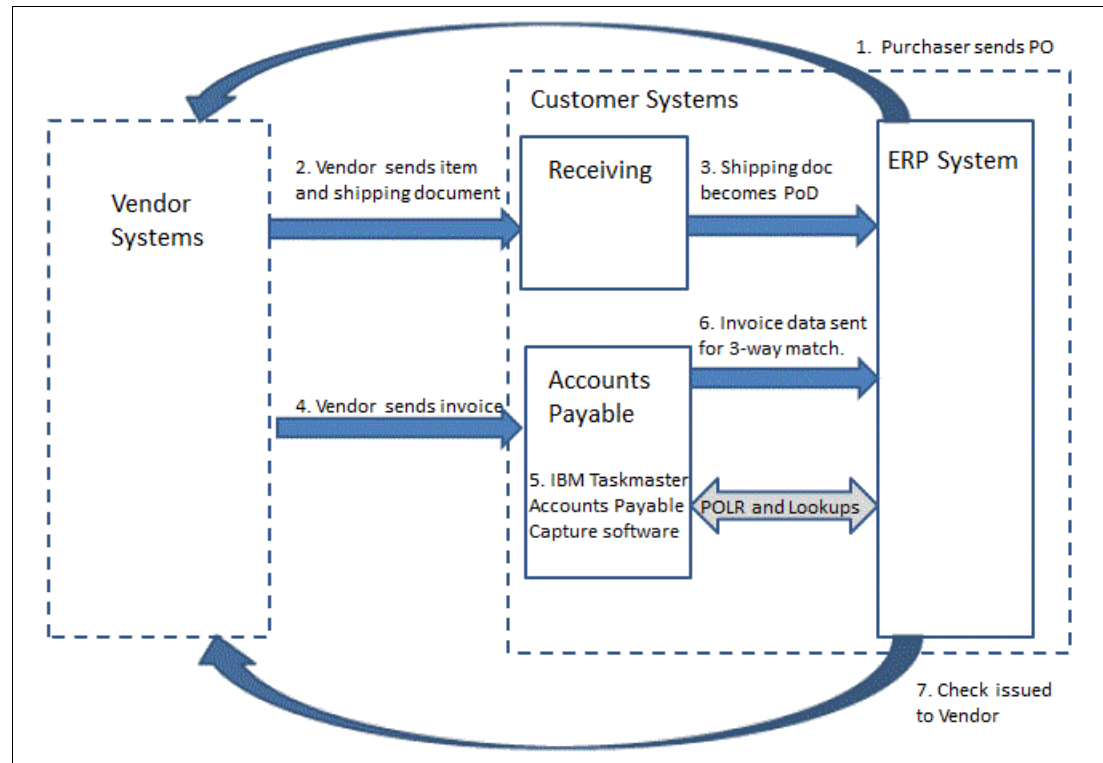


Figure 1 AP process diagram

## Jobs available in the workflow

There are a number of different jobs in the Accounts Payable Capture workflow. Although all of these jobs share common elements, they are different in some way.

Figure 2 on page 5 shows the jobs available in IBM Datacap Accounts Payable Capture.

Workflow	Description
APT	Accounts Payable Technology
Main Job	Scan w/ Scanner
Web Demo	Web Disk Scan
Demo	Scan from Disk
Demo-Dot Matrix	Scan DM from Disk
Main Job-Dot Matrix	DM from Scanner
Web Main	Web w/Scanner
Web Demo-Dot Matrix	Web Disk Scan DM
Web Main-Dot Matrix	Web-DM w/Scanner
Demo-Multipage TIFF	Scan from Disk
Demo-FlexID	Scan from Disk

Figure 2 Jobs in Accounts Payable Capture workflow

Figure 2 shows the jobs in the Account Payable Capture workflow. Key information about the various jobs is as follows:

- ▶ Jobs starting with the term *Web* use a web browser to scan and upload, but the remaining tasks in the job are identical to their Datacap Desktop workflow counterparts.
- ▶ Jobs containing *Demo* use a virtual scanning technique.
- ▶ Jobs called *Main* are used to drive a physical scanner. You cannot set up the Datacap Desktop tasks without a scanner attached to the machine and drivers to operate it. By default, the Scan task is not present. Instead, these jobs begin with *Batch Profiler* but require you to add a physical Scan task as a batch creation task and as the first task in the job.
- ▶ Any jobs that end with *FlexID* have an optional FlexID task before Batch Profiler to manually identify key pages in the batch.
- ▶ The jobs that end with *-Dot Matrix* are identical to jobs with a similar name in all respects except for the name of the job itself. When a batch is processed, during the recognition phase, the job name is queried. If the job ends with “-Dot Matrix”, the recognition engine is configured, using rules, for dot-matrix recognition.
- ▶ Any jobs that end with *-Multipage TIFF* are also identical to the jobs with a similar name, aside from the name itself. At run time, the name of the job is queried to enable rules that do page identification based on the structure of the multipage TIFF input files.

Using nearly identical jobs with different names is useful for applications where minor changes in the execution of rules can be achieved by using different jobs.

## When each task profile gets executed

In most cases, task profiles are run by tasks, but Accounts Payable Capture pioneered the use of running task profiles, on demand, from a Verify panel. By using rules, which you run by clicking a button in a Verify panel, you can reuse task profiles in multiple places during the execution of your application. You can also have a single codebase for most of your application. Calling rules from a Verify panel is preferable to coding and maintaining the same functionality in the panels themselves.

Figure 3 shows the task profiles used in Accounts Payable Capture.

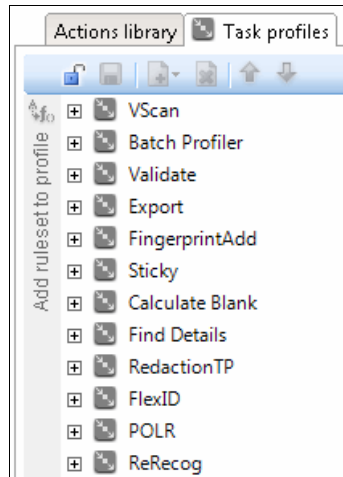


Figure 3 Task profiles in Accounts Payable Capture

The first four task profiles are run by workflow tasks. Except for the Validate task profile, the task profiles are named the same as the tasks that call them.

The Validate task profile is called from a data entry panel during the Verify task.

FingerprintAdd is called when you add a fingerprint manually on the Zones tab in Datacap Studio. With an Accounts Payable Capture workflow, you do not do this often. Instead, the Accounts Payable Capture workflow can add fingerprints to the system automatically, and with greater accuracy, than is possible with the manual interface in Datacap Studio.

Sticky is called from the Verify panels when the Verify panel detects a <New> fingerprint that it has learned zones for, from a previous image in the batch.

CalculateBlank and Find Details run from buttons on various Verify panels.

ReRecog allows the operator to send a page or multiple pages back for recognition in a different language. This action is useful if one or more pages are in an unexpected language and automatic recognition fails.

## A walkthrough of the task profiles

The rest of this paper takes you through the task profiles of the Accounts Payable Capture workflow, as used in IBM Datacap Accounts Payable Capture at the time of writing.

Before you attempt the walkthrough, you must understand the concepts of rulesets, rules, functions, actions, how they are attached to the Document Hierarchy (DCO), and how they are run.

This section highlights the following task profiles:

- ▶ The VScan task profile
- ▶ The Batch Profiler task profile
- ▶ The Verification process
- ▶ The Export task profile

## The VScan task profile

The VScan task profile brings documents into the system programmatically, without user intervention. Other methods exist for bringing documents into the system in an Accounts Payable Capture workflow as well. For those methods, the jobs listed as *Main* require the use of a physical scanner.

Other common Accounts Payable Capture tasks, although not used in the foundation application at the time of writing, pull documents from an email system or a fax server.

**Tweak your application using the VScan ruleset:** Set up, develop, and tweak your applications using the VScan ruleset. When tweaking the system for optimal performance, it is advisable to process the same images a number of times so that you can tell if the modifications you make have the wanted effect. This approach is preferred to running multiple batches through a scanner, where each page might be scanned in slightly differently.

The VScan task profile runs in the Demo jobs that are not designated as *Web* in the Accounts Payable Capture workflow (Figure 4).

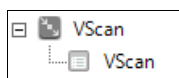


Figure 4 The VScan task profile

By default, the VScan task profile contains one ruleset, that is also called VScan. It is common to add additional rulesets to this task profile if your input images are PDF, JPEG, Grayscale, Color, or some other image type that must be converted to bi-tonal TIFF. If you place the convert actions in the Batch Profiler task profile, rolling back batches to the start of Batch Profiler becomes an issue because the images try to go through the conversion process again.

### The VScan ruleset

The VScan ruleset, shown in Figure 5 on page 8, reads single or multipage TIFF files from the input directory.

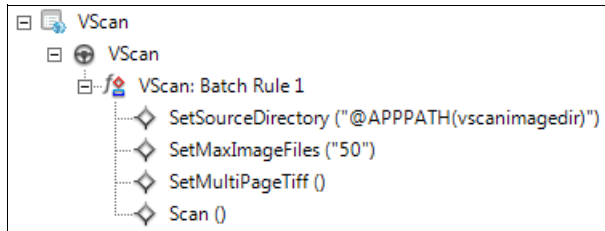


Figure 5 The VScan ruleset

SetSourceDirectory indicates that VScan must look for images in the input folder specified in *VScan source folder* under **Datacap Application Manager** → **APT** → **Application settings** → **Main**. It is preset to accept a maximum of 50 images and to burst any multipage TIFF files into single-page TIFF files.

## The Batch Profiler task profile

The Batch Profiler task profile is the main background processing task profile. It is used in every job in the Accounts Payable Capture workflow and performs all processing between image acquisition into the system and the verify process.

A few of the rulesets, such as Purchase Order Line Reconciliation (POLR) and VendorNumLookup, are specific to invoice processing. They might not be needed in other learning applications.

Figure 6 shows the ruleset composition of the Batch Profiler task profile.

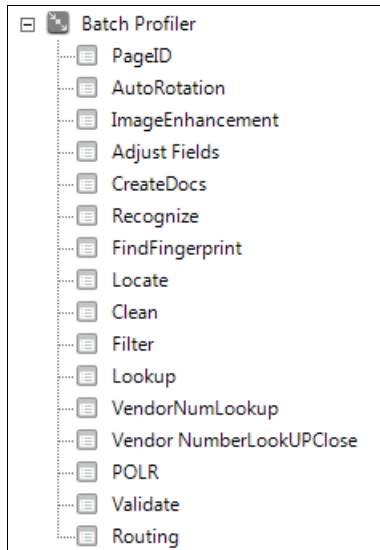


Figure 6 The Batch Profiler task profile

## The PageID ruleset

Most Datacap applications that process structured documents, such as single-page forms, use fingerprinting as the primary page identification technique. However, in situations where a batch might contain several different, multipage documents, and where it is not known whether a piece of data will be found on the first page or on a subsequent page, a different approach for page identification is required.



Learning applications usually have three working page types:

- ▶ Main\_Page (required)
- ▶ One or more optional Trailing\_Page types
- ▶ Attachment page types.

Figure 7 shows the document structure in Accounts Payable Capture.

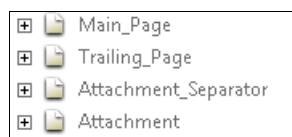


Figure 7 The document structure in Accounts Payable Capture

Accounts Payable Capture uses page identification actions from a library called *PageID.rrx* in the *rules* directory of Accounts Payable Capture:

- ▶ “Identification by barcode separator document” on page 9
- ▶ “Identification by filename variable” on page 9

### **Identification by barcode separator document**

Batches containing one or more single-page invoices need minimal batch preparation. However, batches containing one or more multipage documents need a barcoded separator sheet placed on top of each document. Usually these barcoded sheets are printed on light-colored paper so that, after scanning, the sheets can be located easily and removed for reuse if wanted. An additional barcoded separator sheet must be included between the main document and any attachment pages.

If a batch contains both single-page documents and multipage documents, the single page documents must be scanned first. After the application encounters a document separator, all subsequent documents in the same batch must also use document separators.

### **Identification by filename variable**

Documents that are ingested electronically, for example using the VScan task, do not need separator pages. These documents are typically received as multipage PDF or TIFF files and are identified using a variable containing the name of the file. At run time, when a batch is processed, a variable is placed at page level in the data structure, indicating its origin. This variable is called *ScanSrcPath*.

The *PageIDbyVariableChange* action takes advantage of this feature by using three parameters to name the pages in the batch. The first parameter is the variable to “watch” for change, *ScanSrcPath* in this case. The second parameter is the page type you want to assign to the first page after the watched variable changes. The third parameter is the type you want to use for the remaining pages in the batch until *ScanSrcPath* changes again, indicating the beginning of a new document.

For example, consider an email message that contains three TIFF files. At the batch level, you place the *PageIDbyVariableChange()* action with the appropriate parameters:

```
PageIDbyVariableChange(“ScanSrcPath,Main_Page,Trailing_Page”)
```

Each TIFF file is a multipage invoice. When the images are processed, the TIFF files are separated into individual TIFF files and each page is assigned a *ScanSrcPath* variable containing a reference to the source image. If the email contained two three-page TIFF files, you now have a batch with six images, three of them with a *ScanSrcPath* variable referencing the first multipage TIFF and three images referencing the second multipage TIFF.

The PageIDByVariableChange action processes the batch. Therefore, when the first source image is referenced by the ScanSrcPath variable, it names the page Main\_Page. All pages until the ScanSrcPath variable changes are given the Trailing\_Page page type.

Figure 8 shows how these actions are defined in the PageID ruleset.

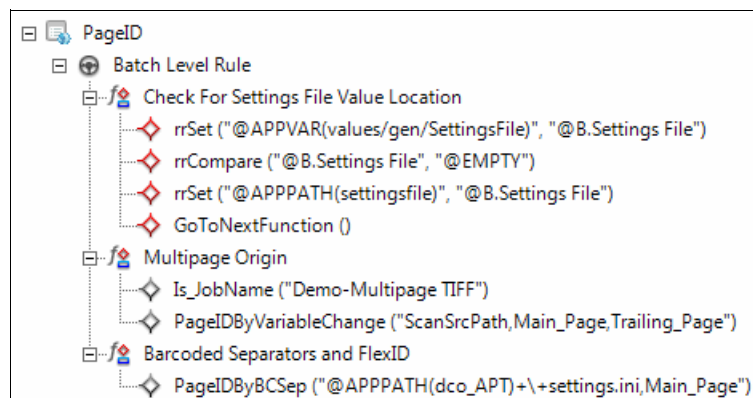


Figure 8 The PageID ruleset

The PageID rule contains a single batch-level rule. When rulesets start processing the DCO, every image in the batch is assigned a batch-level object with a child page of type *Other*. When VScan runs, it places the ScanSrcPath variable on each page containing the name of the source TIFF file. In a case where VScan breaks out images from a multipage TIFF or PDF file, several images in a row can have the same ScanSrcPath variable.

The Multipage Origin() function contains an action, Is\_JobName(), which checks which job name you chose when running the batch. If the job name is *Demo-Multipage TIFF*, this action returns *true* and the PageIDByVariableChange() is run, querying the ScanSrcPath on each image and naming the pages accordingly.

When the ScanSrcPath variable changes from one image to the next, the image with the new ScanSrcPath value is named according to the second parameter of the PageIDByVariableChange() action. In this case, it set assigns Main\_Page as the page type. For every subsequent page in the batch containing the same value in ScanSrcPath, the images get the type specified in the third parameter, Trailing\_Page in this case.

If you are not running the Demo-Multipage TIFF job, the Multipage Origin() function returns false and the function Barcoded Separators and FlexID is run instead. This function contains one action, PageIDByBCSep(), which reads the settings.ini file and uses that information to set the types on all the pages.

As mentioned in section “Identification by barcode separator document” on page 9, you only need barcode separators in multipage documents in the batch. Single-page documents are placed at the start of each batch and get the page type specified in the second parameter, Main\_Page in this case.

When PageID() is complete, all pages must be named. If a page is not typed as *Other* going into the PageID ruleset, it is not renamed. This is because FlexID might have named the pages before PageID runs, and PageID must not overwrite what an operator has specified as the page type. However, if FlexID has run, the actions set the type on any *Other* page that has not been renamed in FlexID according to the *PageID\_LastType\_ThisType* section in the settings.ini file.

## The Image Enhancement ruleset

When Image Enhancement is used in the Accounts Payable Capture workflow, it follows PageID. With form-type applications where fingerprinting is often used for page identification, Image Enhancement runs on every page before the fingerprint match is attempted. When PageID is run before Image Enhancement, you can be more aggressive with the line removal process because barcodes have been read and processed by that point. However, particularly for invoices, you need to be less aggressive on the despeckling process because you need to keep the decimal points in the currency fields if possible.

To define the rules, we use Datacap 9's compiled rulesets capabilities, which provides a GUI-based method to quickly configure, test, and save rulesets (Figure 9).

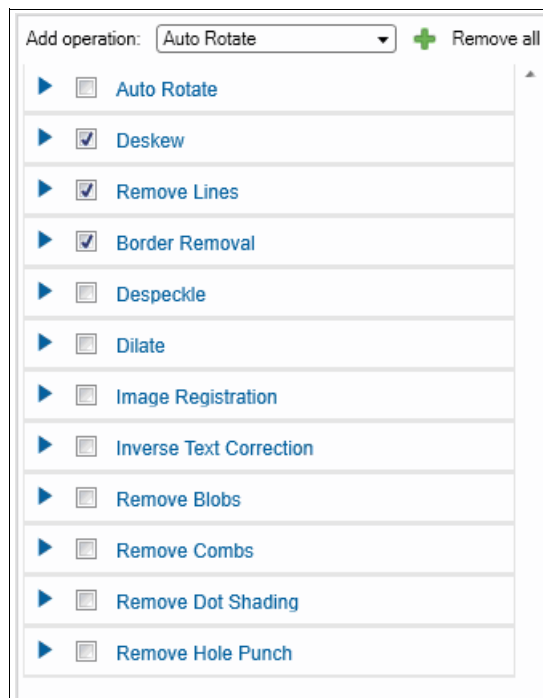


Figure 9 Settings for the Image Enhancement ruleset

## The AutoRotation ruleset

The AutoRotation ruleset automatically rotates the images that it processes. It creates a CCO file to do its work, unless one was already created. The CCO file is created before the image is deskewed. Therefore, you do not want to use this CCO file for fingerprint matching. You can create another one during the Recognize ruleset. Figure 10 on page 12 shows the AutoRotation ruleset.

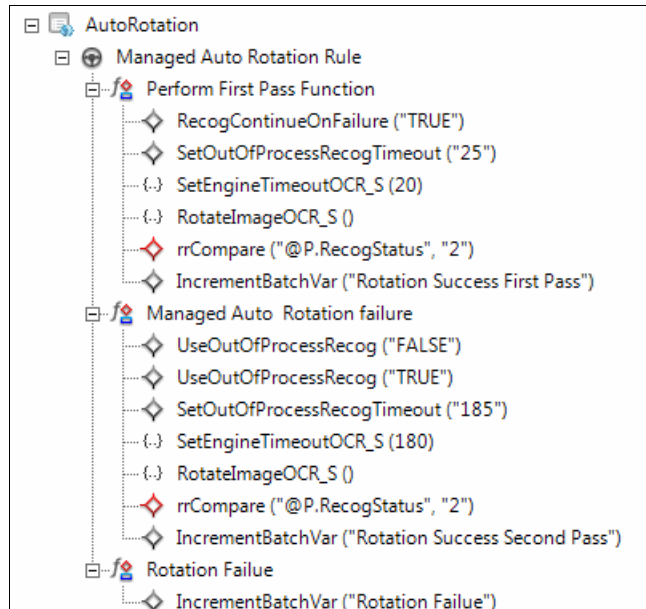


Figure 10 The AutoRotation ruleset

## The CreateDocs ruleset

The CreateDocs ruleset in an Accounts Payable Capture workflow is the same as it is in other workflows. At the batch level, you run the CreateDocs ruleset and at the page level, on pages that contain fields, you run CreateFields.

Figure 11 on page 13 shows a document structure created by the CreateDocs ruleset in the Accounts Payable Capture workflow. When using the CreateDocs ruleset on the Main\_Page object, it always sets it as the first page of a new document called the *Invoice*. The Invoice document can contain all of the other page types that are set in PageID, except for a Document\_Separator page. When found, those pages are placed in a Separator document. No additional processing is done on Separator documents in Accounts Payable Capture. The process ignores the Document\_Separator images from the batch, without removing them, leaving your audit trail intact.

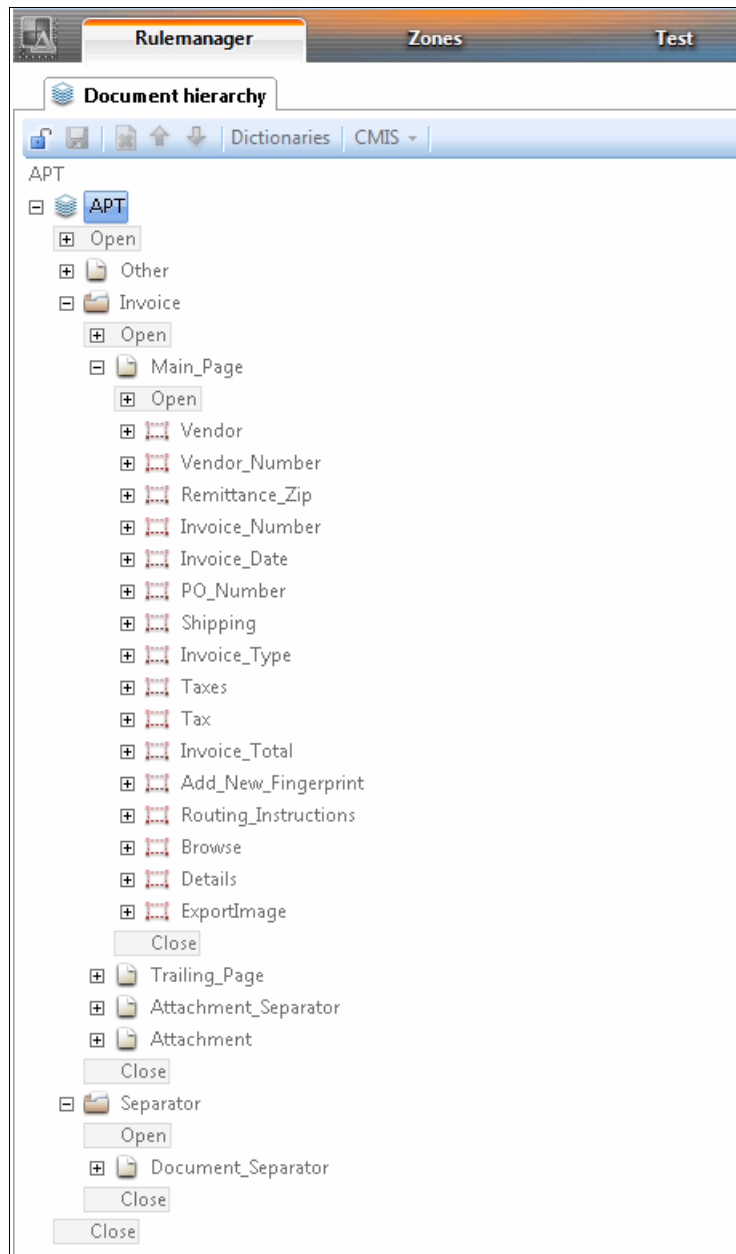


Figure 11 Accounts Payable Capture DCO

## The AdjustFields ruleset

The AdjustFields ruleset creates a document structure called *Browse* so that users have a structure that spans each page of the invoice, and you can browse between them when using the Datacap Web client. The AdjustFields ruleset creates a line-item child, called *PageNo*. *PageNo* contains a subfield named *TIFF*, which is placed in the upper left corner of each page and contains the name of the TIFF image containing the field.

When you are in the Verify panel and you click a field, it automatically shows the page on which the field is found. If you need to move to a page to click data, the thin client does not have buttons to make such a move. The Browse structure places a field and associated buttons so you can move back and forth between pages of the invoice. Figure 12 on page 14 shows the page browse buttons.

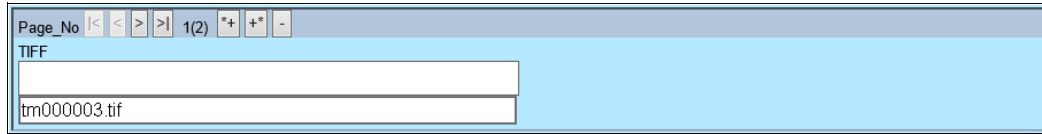


Figure 12 Page browse buttons in thin client

**Note:** At the time of writing, Accounts Payable Capture is being ported to work with Datacap Navigator, the IBM Content Navigator-based thin client released with Datacap 9. Therefore, in this paper, the term *thin client* refers to the Microsoft Internet Explorer web client at <http://server/tmweb.net/>.

Figure 13 shows the implementation of the AdjustFields ruleset. The Add Page\_NoReferences runs on the Browse field on Main\_Page.

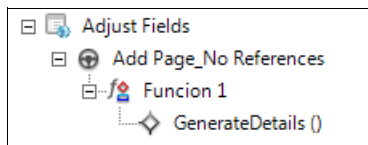


Figure 13 The AdjustFields implementation

## The Recognize ruleset

In the Recognize ruleset, a number of job-specific actions occur. This ruleset also provides a technique for capturing statistics in the runtime DCO.

Figure 14 on page 15 shows the rules, functions, and actions of the Recognize ruleset used in Accounts Payable Capture. The single page-level rule is attached to the Main\_Page and Trailing\_Page objects. The Attachment pages are not recognized.

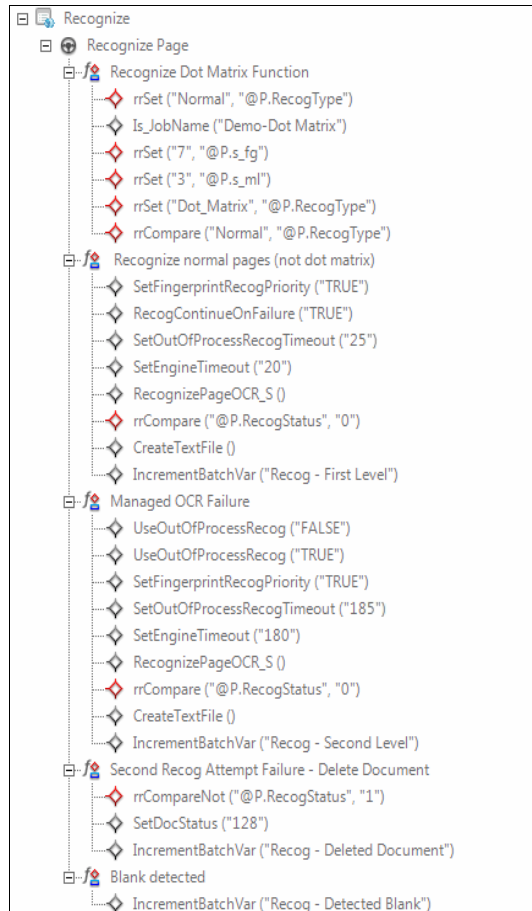


Figure 14 The Recognize ruleset

The first function, the Recognize Dot Matrix function, always returns *false*. The first action, `rrSet()`, creates a batch-level variable called *RecogType*. The action sets its value to *Normal*. The second action, `Is_JobName()`, checks the job name, which is Demo-Dot Matrix in this case. If this name is not the current job name, the action fails, and the next function is started. If *true*, the action adds variables to the page and sets the values of those variables to inform the Nuance recognition engine to use the dot-matrix settings. It then sets the batch-level *RecogType* variable to a value of *Dot Matrix*, and the last action, `rrCompare()`, then returns a value of *False*.

You might need to copy this function if you have more than one job that you want to use for Dot Matrix. You need a function before the Recognize Normal Pages (not dot matrix) function for every job that you add for Dot Matrix recognition. For example, if you want to add a Main Dot Matrix job, you copy the top function and place the copy above the Recognize Normal Pages function. Then, you change the `IsJobName` parameter to the name of your new job.

The second function, which is the Recognize Normal Pages function in Figure 14, shows the first attempt at managed recognition. With managed recognition, the recognition engine is placed outside of the current thread and monitored by Datacap. When recognition occurs, if the recognition engine is successful in recognizing the image, it returns the data directly to the main application thread for further processing. If it fails or times out, a number of additional attempts are made, with longer timeouts, before an error is returned.

Recognition begins with the `SetFingerprintRecogPriority()` action set to *true*. By setting this action, `RecognizePageOCR_S` replaces any existing CCO with a new CCO that is created by

the recognition process. This action is necessary because RotatImage was used in the ImageFix ruleset. It creates a CCO with the AnalyzeImage method and was created before the image was deskewed. This action essentially throws away the CCO and makes a new CCO based on where words and lines were recognized by the recognition engine. This action provides a more usable CCO for locating data than a CCO that was created with AnalyzeImage.

The RecogContinueOnFailue() action is then set to *true*. This way, if a problem image is found in the batch, the batch can continue after you detect this problem and reset the engine.

The next two actions, SetOutOfProcessRecogTimeout() and SetEngineTimeout() actions, specify the timeouts to use. A monitored thread is created that is set to automatically shut down after 25 seconds if recognition is unsuccessful. The recognition engine itself is created in that second thread and has its internal timeout set to 20 seconds. In most cases, a page is recognized quickly, usually in 2 seconds or less. However, if a problem image is encountered, the recognition engine can detect this image based on the time it takes to recognize it. Failing that test, the thread expires if the recognition engine hangs and is unable to monitor itself.

The RecognizePageOCR\_S() action does the recognition. It uses variables that are created with the engine setup on the **Zones** tab in Datacap Studio and, possibly, the Dot Matrix variables. It tries to recognize the page, write the CCO, and return a status.

If the status it returns is 0, everything is successful. A text file is then created in the batch directory with the createTextFile() action. This text file is for observation and troubleshooting only. It is not used elsewhere in the process.

Finally, a batch-level variable is created or incremented with a special action called the *IncrementBatchVariable()* action. This action helps to capture statistics and place them at the top of the runtime page file so that you can quickly look at the page file. This technique is useful if you have several branches processing your images because it counts the number of times that a path is taken. If the variable does not exist, it is created with a value of 1. If the variable does exist, the value is incremented by 1.

The Managed OCR Failure function is a copy of the function before it tries recognition a second time with longer timeouts, but after resetting the engine. The IncrementBatchVar() action logs in the DCO when the managed failure occurs.

If the engine fails a second time, the Second Recognition Failure function deletes the document. However, if the recognition engine returns a value of 0 for RecogStatus, the Managed OCR Failure completes. If the recognition engine returns a value of 1 for RecogStatus, the recognition engine detects a blank page.

If the page is not blank, we have tried to recognize it twice, once with a 20-second timeout, and another time with a 3-minute timeout. In this case, mark the page for deletion, which also deletes the document it is in. This action does not delete the document. Instead, it marks the document so that you do not attempt to process it further. A well-designed system notifies someone about these deleted documents. Accounts Payable Capture does this notification in the Export task.

## The FindFingerprint ruleset

Fingerprinting on the Accounts Payable Capture workflow is only done on the Main\_Page. The purpose of the Fingerprint ruleset is for automatic identification of the specific vendor who sent the invoice. It also provides the offset that is needed to apply to zones to compensate for differences in the scanning positions.



If the Accounts Payable Capture application is being used to capture documents from tens of thousands of vendors, use the Fingerprint Service. The *Fingerprint Service* is a web server that stores the CCOs for all active fingerprints in memory. Without it, the CCOs must be loaded from the network every time a batch is processed, which can add significant time to the background processing.

The purpose of the FindFingerprint ruleset is to ensure that every Main\_Page has a fingerprint TemplateID. If no match is found with an existing fingerprint, a new fingerprint is created automatically. Any new fingerprints created this way do not have zones defined on them and data must be extracted by keyword or regular expression before a data entry operator views and processes the invoice. Later in the “The PreExport ruleset” on page 35, we run rules that save any zones that are identified for these new fingerprints. This way, the next time that this invoice is encountered, the data can be found by using zones.

Figure 15 shows the FindFingerprint ruleset that is used in an Accounts Payable Capture workflow.

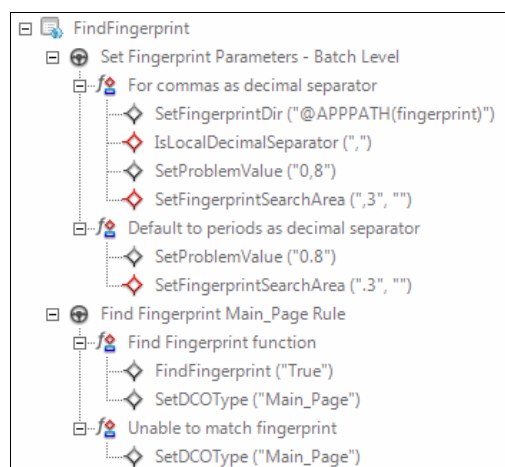


Figure 15 The FindFingerprint ruleset

The batch-level rulesets the fingerprint directory from the App Service setting. New fingerprints are stored in this directory if they are created. The paths to the existing fingerprints are stored in the Fingerprint database.

**Moving the Fingerprint database from one system to another:** With a learning system, such as Accounts Payable Capture, use care when moving the Fingerprint database from one system to another. You must ensure that the paths are correct and that you do not lose entries when moving.

Consider what might happen if you copy your entire system from production to development, work on the system in development for a few days, and then try to copy the entire system back. You might lose any new fingerprints that were added to the production system while the copy was in development. In general, do not move the fingerprint database from one system to another.

In the batch-level rule in Figure 15, the first function, *For commas as decimal separator*, checks the locale of the machine that is processing the fingerprints. It ensures that the floating point values from SetProblemValue and SetFingerprintSearchArea use the correct decimal separation character in their parameters.

Normally, we want to look at the top 30% or so of the current image to compare it against our fingerprint library for matching. The ProblemValue, by default, is set to 0.8. Decreasing this

value increases the chance that the fingerprint (incorrectly) matches a fingerprint from another vendor, creating fewer fingerprints in the system overall. Increasing this value gives a more precise match, creating more fingerprints in the system and additional work for data entry operators in zoning these additional new fingerprints.

Adjust this setting only after a discussion with the users about the effects they can expect by changing this value. A value of 0.8 often provides the correct balance.

If a mismatch does occur, data entry operators can create a fingerprint with the click of a button, or in the case of the web client, by choosing **YES** from the Add New Fingerprint list. If more than one fingerprint exceeds the ProblemValue setting, the fingerprint with the best match is returned by the FindFingerprint action.

In the page-level *Find Fingerprint Main\_Page* rule, the SetDCOType action ensures that the FindFingerprint action does not change the page type (which is set in the PageID ruleset) to another value.

## The Locate ruleset

In the Locate ruleset, data is pulled from the CCO created in the Recognize ruleset. This ruleset uses a number of techniques for retrieving data. We explore each technique.

The Locate ruleset is run as part of two different task profiles, Batch Profiler and Sticky. Figure 16 shows how the Locate, along with CheckForSticky, Clean, DynamicDetails, Filter, and Validate rulesets are used in the Sticky task profile, which is called during the Verify process.

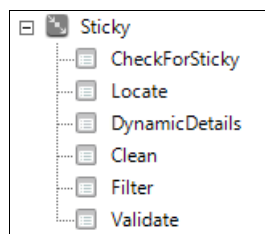


Figure 16 The Sticky task profile

Reusing rulesets in this way reduces the maintenance of the system. However, it does make each ruleset slightly more complex because it must test when it is being run so that it knows how to handle its operations when processing or reprocessing a document.

## Sticky fingerprints

“Sticky” fingerprints is a technique for “saving” a new fingerprint’s location information at verify time so it can be applied immediately to other images in the batch that match the same (new) fingerprint.

For example, when an operator encounters a new image with no matching fingerprint, a new fingerprint is automatically created and he or she extracts, or indexes, data using Click N Key capability. This provides location information for that fingerprint.

When a second image of the same type, from the same vendor, is encountered, a *Sticky Available* button is displayed. When the operator clicks that button, the location information from the previous invoice is applied immediately to the current one. The CCO is searched for the header and line-item data, and the document is processed without further operator interaction.

Figure 17 on page 19 shows the document-level and page-level rules of the Locate ruleset.

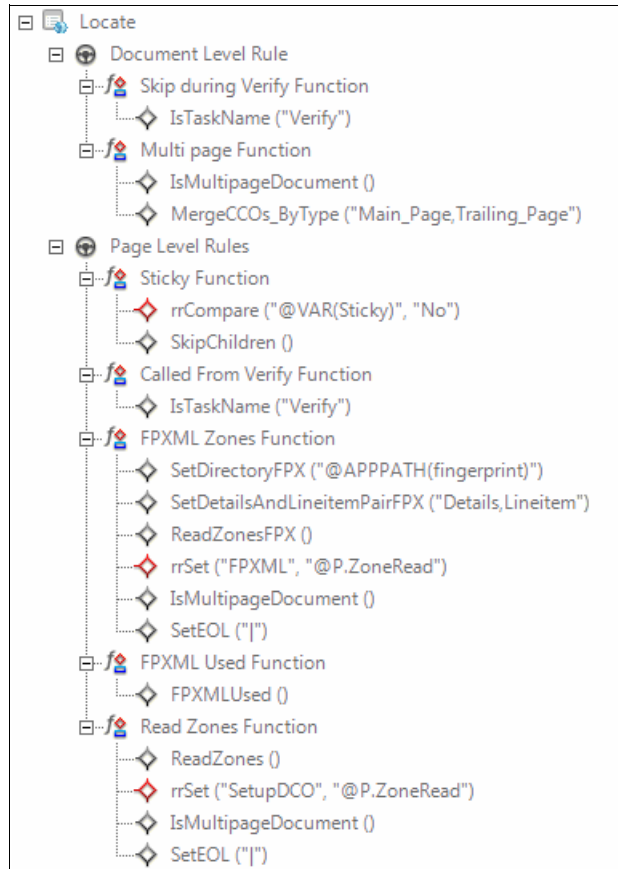


Figure 17 Document and page-level rules in the Locate ruleset

The document-level rule, *Document Level Rule*, creates a multi-CCO file (MCCO) by combining all of the CCOs from the Main\_Page and any Trailing\_Pages and replacing the CCO of the Main\_Page with the larger, combined one. This way, the entire document can be searched for data at the same time.

When making an MCCO, you only do it once and only during the Batch Profiler task profile. To prevent merging CCOs a second time, there is a check in place, on the document-level rule, to make sure merging does not take place during the Verify task. Running this action more than once duplicates data from the Trailing\_Pages to the merged CCO, potentially doubling the data found when we search for it.

In the first function, *Skip during Verify Function*, we check to see if we are running under the Verify task. If we are, the function returns *true* and does not run the second function that creates the MCCO.

In the second function, the `IsMultipageDocument()` action determines if the document contains more than one page. If it does, it makes an MCCO for you.

For the page-level rule, we are trying to read any zones, if they exist, that are associated with the fingerprint. Similar to the document-level rule, we must check to see when the Locate ruleset is running so we do not reprocess documents that do not need it.

The first function checks to see if we are in a situation where sticky fingerprint technology is useful. See the Sticky task profile (see Figure 16 on page 18). The first ruleset in the Sticky task profile sets a variable named *Sticky* that indicates whether we need to reprocess the runtime document with Locate rules. If the value of the Sticky variable is *No*, the

SkipChildren() action causes the ruleset to stop running for all children of the page, meaning that the fields will not be searched for data.

Regardless of whether Sticky is set to *Yes* or *No*, we do not want to attempt to read the zonal information when we run from the Verify task. This is because the first ruleset in the Sticky task profile also sets the zones based on the previous document in the batch. The net result of the first two rulesets is that, when running under Sticky, the zonal data is never read by the Locate ruleset. Depending on whether the Sticky variable is set to *Yes* or *No*, the field-level actions in the Locate ruleset will or will not run.

The remaining functions on the page-level rule must only run during the Batch Profiler task profile. Again, because there are two different places where zones can be stored (FPXML or the Setup DCO), we must check to see which actions are appropriate for reading the zonal information.

Reading zones with the FPXML method requires us to set the fingerprint directory where the FPXML files are kept. To read FPXML with detail lines defined, we must inform the ReadZonesFPX() action of a detail structure so that it can read and apply the detail line zones correctly. This process is done by using the SetDetailsandLineitemPairFPX() action. You do not have to specify the Browse structure here. Such zones are set programmatically in the AdjustFields ruleset, run previously in the Batch Profiler task profile.

FPXReadZones returns FALSE if it cannot find or read an FPXML file. If it is a multipage document, set an EOL character for the MCCO to process correctly. If the FPXML is read, a variable is set at the page level so that we know the method by which the zones are read. Because there are two actions that can return FALSE in this function, we must check this variable in the next function to see if the zones are read in correctly. If the ReadZonesFPX() action returns FALSE, the variable is not set, and it attempts to read zones from the Setup DCO.

The rest of the Locate ruleset pulls data into the fields by using the following methods:

- ▶ Populating data that is normally best found zonally
- ▶ Finding data that floats around
- ▶ Searching for and populating the detail lines

### ***Populating data that is normally best found zonally***

Using the first two methods listed above, Accounts Payable Capture checks zonally and uses keyword searches. The order in which the methods that are used is important: Figure 18 on page 21 shows how we find data when zones are preferable for locating the data. We use this method when data is in a predictable place on the page.

As with the previous rules in this ruleset, we must check which task we are running under. The first function, *Called from Verify Function*, checks to see if we are running under the Verify task. If we are, then it loads the CCO dynamically and attempts to populate the data from a zone using PopulateZNField(). If we are not running under the Verify task, the CCO has already been loaded and we can immediately run the PopulateZNField() action. If a zone is defined for the field and the zone contains data, the data in that position is pulled into the field. The rule for that field is finished, and we can move on to the next field.

If no zone is defined (for example, when we have a new fingerprint) or if the zone contains no data, we use Locate actions to try to find the data programmatically.

The first step in this process is to find data using a keyword or regular expression. Although regular expressions are a powerful method of finding data in general, most of what we capture from invoices is not sufficiently unique to justify finding it using a regular expression. The exception is a Purchase Order (PO) field. If a company uses a number that is fairly

unique, such as the letters PO followed by eight digits or something of that nature, consider using a regular expression to find that data.

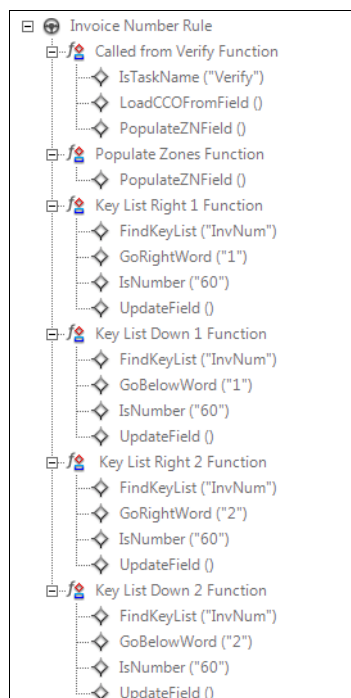


Figure 18 Finding data with the zonal method preferred

With the keyword search, a key file is placed in the dco\_<AppName> directory listing the keywords that might be used as labels around the data that you are searching for. Accounts Payable Capture ships with a limited set of keywords in the .key files. Add additional keywords as necessary.

After a keyword is found, we move one word to the right to see if that word contains data that is appropriate to the type of data we are looking for. If it does, we do an UpdateField action, which completes the process. The rest of the functions in the rule operate identically, but look at different words based on their location related to the keyword that was found.

This method is used for the majority of fields in a document where the data for that instance of a document is in a static location (because we are checking for the zone first).

### ***Finding data that is not in a fixed position***

Some data, such as the Invoice Total, Tax, and Shipping, and so on, are normally found on the last page of an invoice. Invoices might have a variable number of pages. One day you might get an invoice from a vendor that has one or two line items and everything fits on a single page. The next day, you might get an invoice from the same vendor that has dozens or hundreds of line items and is many pages long.

For this reason, we use a different method for data that typically occurs below the last line item on an invoice (Figure 19 on page 22).

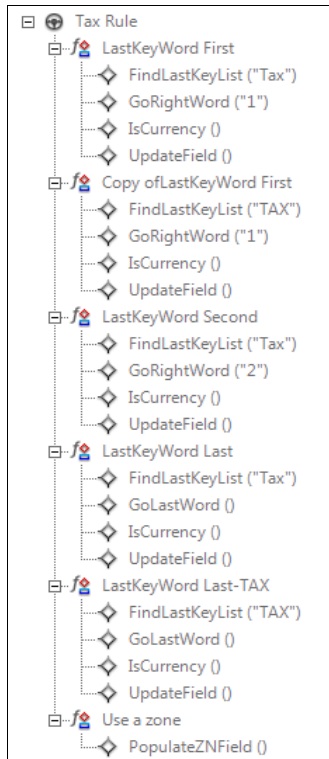


Figure 19 Rule for finding data that is not in a static position on the page

In this case, we look for the last occurrence of a word in a document and then look “around” that word for data that fits the data type we are searching for. If the word is not found, we use the zone as a last resort.

### Searching for and populating the detail lines

The technology includes finding detail lines. However, more is involved in the actual implementation than just the actions that search the CCO for lines as illustrated in Figure 20.

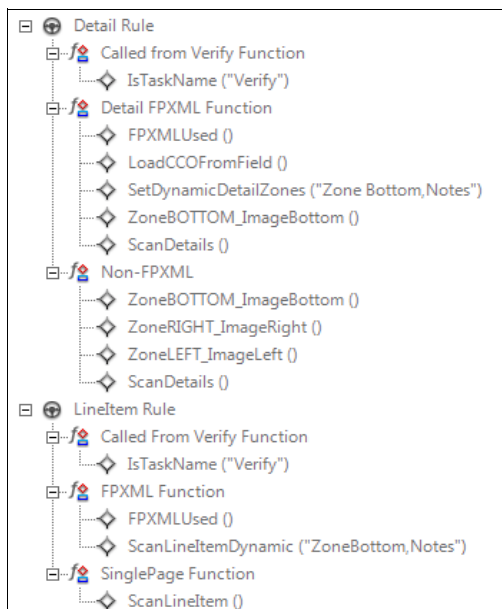


Figure 20 Finding detail lines in the Locate ruleset

As with some of the other rules in the Locate ruleset, we must check if we are running in the Verify task to prevent reprocessing the document unnecessarily. The first function does this check. If you are running under the Verify task, the ruleset does not use these actions to find detail lines.

The second function checks whether we are using FPXML and moves the FPXML zones to the detail structure. FPXML stores zones different from the Setup DCO. If FPXML is used, we must load the CCO again. The next two actions set up the zone for the multipage document, ensuring that the entire CCO is searched for line items, even though it contains a variable number of pages. Regardless of whether FPXML is used, after the zone is set up, ScanDetails creates the detail structure and puts each line item in its own child Lineitem field.

At the line-item level, if FPXML is used, we perform a ScanLineItemDynamic action using the recently loaded CCO. Alternatively, we perform the ScanLineItem function if the zones were read from the Setup DCO.

Finally, each field on the line item is populated with PopulateZNLinItemField.

### **The Clean ruleset**

The Clean ruleset is used for a specific purpose on a limited number of fields in an Accounts Payable Capture workflow. Most field cleaning occurs in the Validate ruleset. When a data entry operator clicks a field, we want to remove unwanted characters and set the data format in fields, such as dates, to a specified format.

However, some fields are used for data lookup before we validate them. Therefore, those fields must be cleaned and potentially formatted using this ruleset. Also, we must check the data in certain line-item fields to ensure that it is the data that we are searching for before filtering the line items.

Figure 21 on page 24 shows the default ruleset for cleaning invoices.

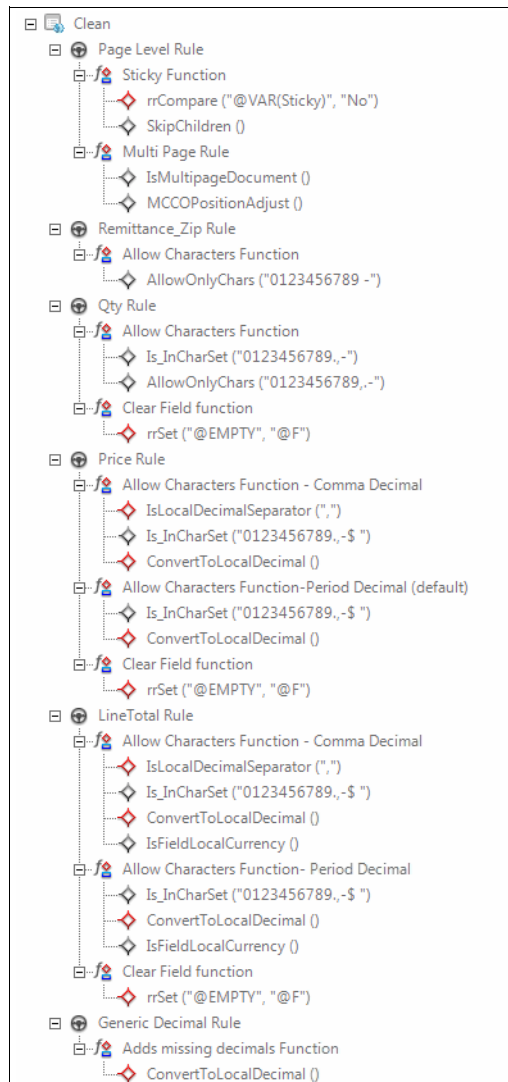


Figure 21 The Clean ruleset

The page-level rule indicates that the Clean ruleset must do nothing if we are running under the Sticky task profile and the current fingerprint is not eligible for sticky processing. If the rule fails, a first attempt is made to adjust the field positions from the MCCO to coordinates that are associated with each page.

Because the ZIP field is used in a vendor lookup, the data is cleaned of any character that is not a number (or potentially a dash, if postal codes contain dashes in your database). This action ensures that we have removed any spaces or extraneous characters that might cause the lookup to fail.

In the line-item fields, we delete any unexpected data. For example, if a Qty field contains the word “Thank,” which it might have captured when reading the bottom of an invoice, this word is deleted. The same is done for Price and Line Total in the detail-level fields.

Avoid the urge to clean fields in this ruleset that are not used in lookups or for detailed line filtering. Otherwise, you have to do it again in the Validate ruleset, which adds to maintenance if the cleaning parameters change.



## The Filter ruleset

The Filter ruleset is used to delete captured line items that do not fit the data types we are expecting.

The Filter ruleset used for invoices in the standard Accounts Payable Capture workflow is shown in Figure 22.



Figure 22 The Filter ruleset

The page-level rule avoids reprocessing line items if we are not in a condition where it is beneficial to do, such as when running under the Sticky task profile.

In the detail field, the CheckSubFields erases all line items that do not have valid data (after cleaning) in two of the three Qty, LineTotal, and Price fields. After cleaning, MCCOPositionAdjust adjusts the raw CCO coordinates for a multipage CCO into coordinates that are adjusted to the single page image from which the data originated.

The lookup ruleset populates the Vendor name in the vendor field. The Vendor name is stored in the fingerprint database. However, the ruleset has an option for pulling this information from a locally stored database. An example is a mobile computer for development that might be disconnected from a network installation. Therefore, the ruleset can still function in a disconnected state (Figure 23).

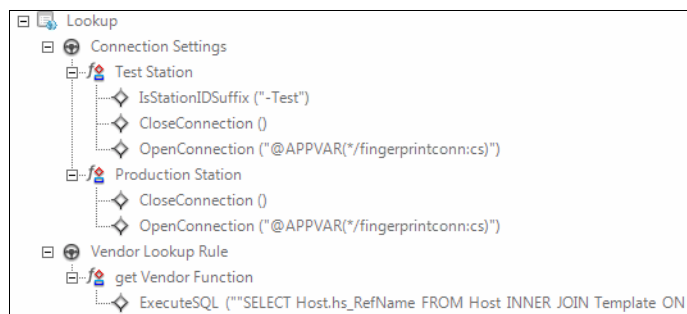


Figure 23 Lookup ruleset populating the Vendor field from the fingerprint database

The IsStationIDSuffix() examines the station ID of the users in Datacap and can conditionally run one of two OpenConnection actions that open a connection to a database. By default, these actions are set to the same value. However, the first function can be altered if you are working on a machine that is sometimes disconnected from the fingerprint database.

The ExecuteSQL action in the Vendor field checks the fingerprint database and retrieves the fingerprint classification. In Accounts Payable Capture, this classification provides the vendor name, but in Flex, this classification provides the document type.

## The VendorNumLookup ruleset

The VendorNumLookup ruleset must be customized when Accounts Payable Capture is deployed at a customer site. It ships with an Access database so that the demos will work.

In an installed system, Datacap Accounts Payable Capture gets the vendor numbers directly from the business application system or from a recent copy of the vendor database obtained from the business application system. The vendor number is not assigned to just a vendor name, but also to a vendor location, because many vendors have different addresses for the business application system to send checks to.

You might want the vendor number to be found based on the PO number. This strategy is also successful if the business application system can supply such data through a lookup.

Because this ruleset must be customized, and a readily available version is identical in structure to the Lookup ruleset, we do not address this ruleset any further in this paper.

## The Vendor NumberLookupClose ruleset

The Vendor NumberLookupClose ruleset falls into the same category as the VendorNumLookup ruleset. The Vendor NumberLookupClose ruleset ensures that you have closed off the connection to the vendor database in the production system.

## The POLR ruleset

The purpose of the POLR ruleset is to try to furnish line item numbers to each of the recognized line items in the invoice. Similar to the previous two rulesets, this ruleset must be customized to pull the line item data from the customer's business application system.

POLR uses settings from the settings.ini file to do the work. In most cases, just altering these settings is sufficient to allow POLR to work at a customer site. Figure 24 lists the pertinent settings from the settings.ini file.

```

If the Purchase Order Line item Reconciliation module is used, the following entries need to be modified
for the customer supplied PO Database tables. The Test entries should point to the Datacap supplied tables.
*****
TestPODSN=Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Datacap\apt\APTLook.mdb;Persist Security Info=False
TestPOLookup=Select (LineNo,QTY,ItemNo,Description) from POTable where PO = ?
*****
The following two vendor references should point to the AP System vendor tables
*****
For the POLookup value, you need to return the 5 items in the order listed.
PODSN=Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Datacap\apt\APTLook.mdb;Persist Security Info=False
POLookup=Select LineNo,QTY,ItemNo,Description,Format(UnitPrice,'.00') from POTable where PO = ?
*****
FingerprintDatabase=Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Datacap\apt\APTFingerprint.mdb;Persist Security Info=False

[POLR]
A value of 1 in the following line item fields denotes that a match of the field is required for POLR to automatch. The
AutoMatch feature requires an exact match of all fields marked with 1 to automatically match the PO lines.
ItemID=1
Qty=1
Price=1

The price tolerance may be adjusted to allow for minor variations in the price based on rounding or truncation of decimals on the invoice.
This tolerance is used for the automatching of line items only (when Price is set to 1).
PriceTolerance = .005

POLR has the ability to write unused PO Lines to the Detail field as variables. A value of 1 enables this functionality. The SeparatorCharacter
value is the delimiter. If Unspecified, then a pipe is used.
WriteUnusedPOLines=1
SeparatorCharacter=|

```

Figure 24 POLR settings stored in the settings.ini file

The first section in Figure 24 on page 26 is the [Database] section. Similar to the Lookup ruleset, POLR can also use a test database when it is being used in a disconnected state. The settings must point to the vendor business application table that contains PO information or the stored procedure that you use to retrieve the line items based on the current PO.

After the line items are successfully retrieved, POLR uses the ItemID, Qty, and Price keywords to determine which fields to use for automatch. In the previous example, lines automatch only if they have identical values for all three fields between the recognized line items and those retrieved from the business application system.

With manufacturing concerns, item prices are sometimes represented by a small fraction of the currency used. Therefore, a PriceTolerance is specified when automatching line items.

You might also want to output a list of any unused line items on a PO so that users can know that the invoice that they are paying does not close out a PO. For this reason, POLR also contains the capability to write the unused DCO lines to the DCO itself. Although the Accounts Payable Capture workflow does not immediately do anything with these line items, they are stored as variables on the Detail field of each Main\_Page and can be exported if wanted. See the details section in Figure 25 to see how unused lines from the PO are stored.

F	Details
	Text value :
	Char confi :
	TYPE : Details
	Position : 0,1097,2552,3295
	STATUS : 0
	Unmatched Invoice Lines : 3
	Unused PO Line 1 : 30  1100-NM Mileage Estimate 170.00
	Unused PO Line 2 : 40 1  Additonal Labor 400.00
	PreVerify Val :
	PreVerify Pos : 0,1097,2552,3295
F	Lineitem0
	Text value :
	Char confi :
	TYPE : Lineitem
	Position : 200,1102,2377,1147
	STATUS : 0
	PO Line Number : 20
	PO Qty : 1
	PO Item ID : 1000-NM
	PO Description : Base Labor
	PO Reconcilled By : auto
	PO Unit Price : 2100.00
	PreVerify Val :
	PreVerify Pos : 200,1102,2377,1147

Figure 25 How POLR stores data in the runtime DCO

The actual PO values are stored for each line in variables at line-item level. The PO Line Number must be exported so that a three-way match can be accomplished programmatically. However, some systems use other criteria, such as the description, when matching lines for a three-way match. If so, consider exporting the line criteria that allows for the most accurate three-way matching. POLR allows the system and data entry operators to reconcile the lines to keep errors to a minimum. The goal is also to supply additional data to the three-way matching system. This way the knowledge workers assigned to that process can concentrate on actual issues (items not received, incorrect counts, different prices) and do not have to match the lines against the PO.

Although POLR runs in the Batch Profiler phase, it cannot do its job if the PO is initially misread. This task profile prematches as many line items as possible before verification. The data entry operator has an advanced POLR user interface to quickly match any line items that could not be matched in the background process.

## The Validate ruleset

The Validate ruleset is used to clean and format data. It is also used to apply business rules to ensure data is the correct length and data type and is mathematically correct. Validation also checks to ensure that the data meets other business criteria before it can be successfully exported.

The Validate ruleset runs in several task profiles. When running in the Batch Profiler task profile on a background machine, before verification, this ruleset formats and checks each field before a data entry operator views the document. The ruleset then marks any problem fields that it finds so the data entry operator knows they need attention.

This ruleset also runs when the data entry operator has viewed a document, made any necessary corrections, and submitted it as complete. Having the business rules in this single code base saves a lot of development and maintenance time within Datacap systems.

The Validation ruleset in an Accounts Payable Capture workflow also employs enhanced error messaging. With this feature, you can write your own error messages, which are displayed to the operator, if the operator submits a form that fails validation.

This ruleset contains many rules, making it difficult to explain rule by rule. Instead, this section shows how different technologies are implemented within this ruleset. Figure 26 shows the page-level rule and a simple field showing Enhanced Error Messaging.

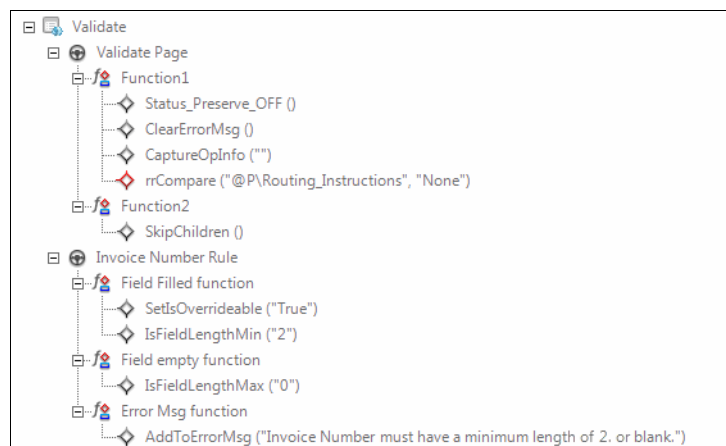


Figure 26 Page-level rule and error messaging in Validate ruleset

In the Validate Page rule, the Status\_Preserve\_OFF() action places the system in a state where the page and every field on the page have a status of 0. At this point in the process, a status of 0 means that all fields as well as the page itself have passed the validation rules. After the validate rules are run on each field, if a field fails validation it is assigned the problem status of 1. The page is then also assigned a status of 1.

Pages with a status of 1 are displayed to a data entry operator, while pages with a status of 0 are not displayed. Fields with a status of 1 are displayed in pink during verification. This way the data entry operator can quickly see, by the color of the field, that it failed the business rule associated with it. More information about status is provided later in this section.

The `ClearErrorMsg()` action is part of the Advanced Error Messaging employed by Accounts Payable Capture. A page-level error message variable that contains text from every field that failed validation is displayed when a data entry operator submits a form that fails any business rules. This action clears that variable before running the field validations.

The `CaptureOpInfo()` action writes the current operator and station information to the DCO for export at a later time, if wanted.

The `Routing_Instructions` field is in the Accounts Payable Capture workflow. Normally this field is set to *None*, but an operator can identify documents that they cannot process for some reason by choosing a predefined routing instruction. The `rr_Compare()` action at the end of the first page-level function (Function1) detects whether this field contains a value such as Delete, Rescan, or Review. If it contains anything other than *None*, the function returns *False*, and the second function (Function2) instructs the system to skip the validation rules on the fields.

For a sample field validation, see the Invoice Number Rule in Figure 27. Each rule can be set to *overridable* or *non-overridable*. Non-overridable fields must be corrected by the data entry operator. Otherwise, they are unable to proceed in verifying a batch.

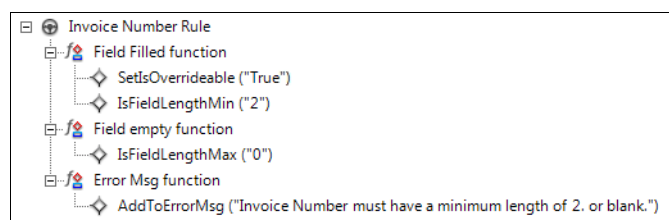


Figure 27 Example of field validation

For this field, a minimum length of 2, with data that is at least 60% numeric, is required according to the first function, *Field Filled function*. If those conditions are met, the rule is finished processing, and the status on the field remains 0.

If this field fails the conditions set in the first function, the second function, *Field empty function*, checks whether the field has a maximum length of 0, meaning that it is blank. If this field passes that function, the rule is finished, and the status on the field returns 0.

If the first two functions fail, the Invoice Number Rule calls the *Error Msg* function, which calls the `AddToErrorMsg()` action. This action sets the enhanced error message by writing to the page-level variable. Because every function failed on the validation, the status of the field is set to 1, meaning it is displayed in pink to the data entry operator. The page status is also set to 1, meaning that the page is displayed to the data entry operator.

Almost every field in the Accounts Payable Capture workflow contains similar validations to apply the appropriate business rules. A number of the fields have additional validation techniques applied to them, in particular localization actions and mathematical calculations. Figure 28 on page 30 shows the localization and mathematical techniques that are used.

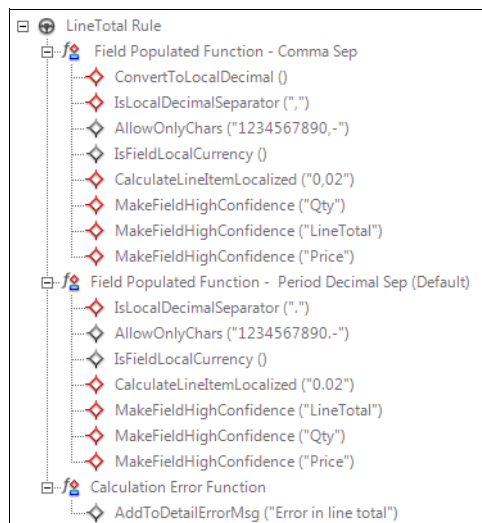


Figure 28 LineTotal Rule for the Validate ruleset

The first action examines data and normalizes it to the decimal separator used on the local machine. Accounts Payable Capture supports both commas and periods as decimal separators. This action examines the data in the field and changes the decimal separator to the same decimal separator set on the machine processing the rules.

Next, the LineTotal Rule checks this separator so that it can properly clean the data of any currency symbols and remove any thousands separators that are present. Depending on the local decimal separator, different characters are allowed.

The LineTotal Rule then checks to ensure that the field is a currency field. It calculates the line item by multiplying the Qty by the Price and ensures that it equals the line total, within the tolerance specified as a parameter. In many applications, having a tolerance is important because the document might limit the number of decimal places that are displayed to fewer than what is required for an exact equivalence match.

If the calculation is correct, the field values must also be correct, even if they contain low-confidence characters. In this application, the fields involved in the calculation are marked as high confidence because we know they are correct based on the calculation.

## The Routing ruleset

In general, the Routing ruleset is used to check a batch for low confidence characters and problem fields to determine whether the batch should go to a data entry operator. However, with the volume of data in a batch of invoices, it is unlikely that the data entry step can be skipped completely. Therefore, the Routing ruleset is used to do final preparation of the batch before a data entry operator sees it. Figure 29 on page 31 shows the Routing ruleset as implemented in the Accounts Payable Capture application.

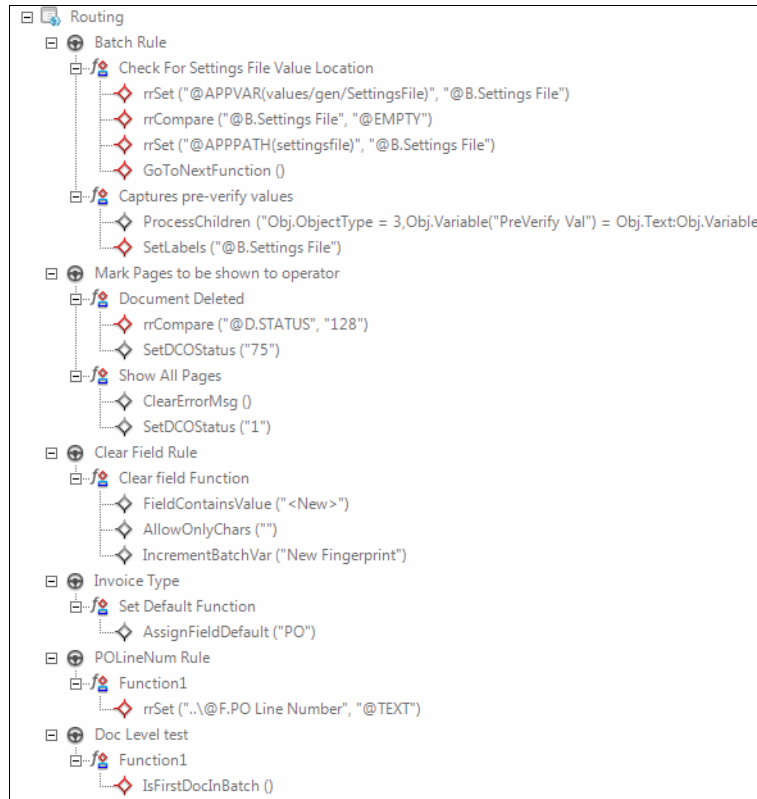


Figure 29 Routing ruleset

The Batch rule uses a ProcessChildren() action to capture the preverify value and position of each field and to store it as a field-level variable. The preverify position variable is used to determine if a field was zoned by the data entry operator. You can use the preverify value to capture statistics on the number of fields that a data entry operator changed, but it is not available immediately.

The page-level rule called Mark Pages marks pages that can be shown to an operator. The first function checks whether the document has been deleted.

Deletion of the document can only happen if a page in the document is not recognized. If the page is not recognized, the page is marked with a status of *deleted*.

All other Main\_Pages are marked with a status of 1, meaning that the data entry operator sees every Main\_Page in the batch. Many organizations prefer carrying out a visual check on every page to make sure that everything is working before exporting the data and the image.

If a document is new to the system, the Vendor field is populated with the value <New> during the Lookup ruleset. The Clear field rule erases this text from the field and leaves it blank. It also increments a batch variable so that you can have a record of how many new fingerprints were in this batch.

The Invoice Type rule defaults the Invoice\_Type field to a value of PO. The POLineItemRule copies the POLR variable for the PO Number to the text property of the appropriate field.

## The Verification process

Several rulesets run under task profiles during the verification process. In verification, task profiles can be set up to run automatically or when a data entry operator clicks a button. This



section highlights the following rulesets that can run under task profiles called by the verification process:

- ▶ The DynamicDetails ruleset
- ▶ The CheckForSticky ruleset
- ▶ The AutoCalc ruleset

## The DynamicDetails ruleset

The DynamicDetails ruleset is a way to find, at verify time, all of the line item fields in a document. You click the line-item subfields of the first detail line on the invoice and click the **Find Details** button. This ruleset sets up the Lineitem and Detail zones automatically.

The DynamicDetails ruleset runs under the Find Details task profile and is called by the **FindDetails** button on the **Verify** tab. Figure 30 shows the Find Details task profile.

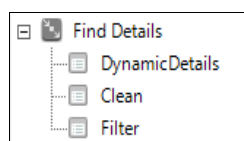


Figure 30 Find Details task profile

The DynamicDetails ruleset (Figure 31) is identical to the rules that are associated with the Detail level and Lineitem rules in the Locate ruleset. However, special actions are needed to read from a CCO that is loaded dynamically at verify time. The actions function identically to their non-dynamic counterparts.

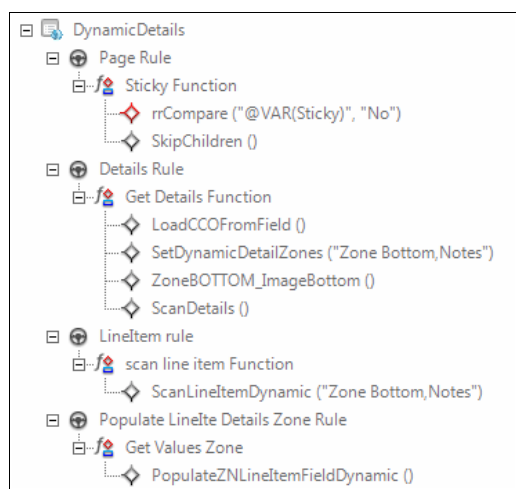


Figure 31 DynamicDetails ruleset

The Clean and Filter rulesets called in the task profile are the same rulesets that are called in the Batch Profiler task profile.

## The CheckForSticky ruleset

The CheckForSticky ruleset runs in the Sticky task profile as shown in Figure 32 on page 33. As explained previously in this paper, you must be familiar with the other rulesets that are called in this task profile.



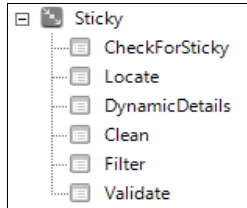


Figure 32 The Sticky task profile

The Sticky task profile runs automatically from Datacap Desktop when it detects that a new fingerprint is being processed and another of the same new fingerprint was processed previously in the same batch. For example, a vendor sends in two invoices with a new format. The first one creates a fingerprint, and the second one matches that same new fingerprint. At verify time, the operator zones the first invoice. When the second invoice is displayed, the Sticky task profile copies and adjusts the zones from the first invoice to the second invoice. Then, it automatically populates with data.

If you are reading this paper from the beginning, you are familiar with all of the rulesets called by Sticky, except for the CheckForSticky ruleset (Figure 33).

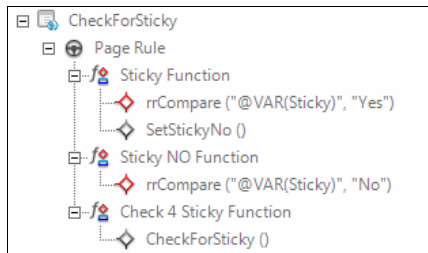


Figure 33 CheckForSticky ruleset

The Sticky variable must be blank when this ruleset is run. The first two functions fail, and the CheckForSticky action runs. This action checks whether any previous documents in the batch can be used to zone the current document. If there are such documents, this action adjusts and copies the zones to the new document. It also sets the Sticky variable to *Yes* or *No* depending on what it detected when it analyzed the batch. If the variable is *Yes*, the other rulesets in the task profile run, and the data is populated. If the variable is set to *No*, the other rulesets do nothing.

The first two functions of this rule are there in case Sticky is run for a second time for some reason. If the Sticky variable is already set to *Yes* or *No* instead of blank, the CheckForSticky rule does not run.

## The AutoCalc ruleset

The AutoCalc ruleset (Figure 34) runs in the CalculateBlank task profile. It is called when the verify operator clicks a button. Because the Qty, Price, and LineTotal fields are mathematically related, this ruleset enters a single missing value if it is detected on a line item.

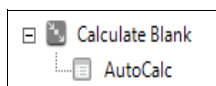


Figure 34 The CalculateBlank task profile showing the AutoCalc ruleset

The CalculateBank task profile calls a single ruleset to do the analysis and automatic correction of a blank Qty, Price, or LineTotal field on a LineItem (Figure 35 on page 34).

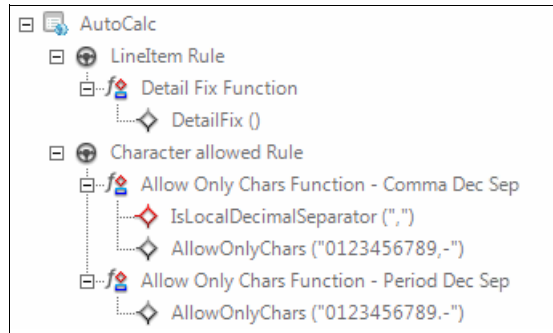


Figure 35 The AutoCalc ruleset

The *Character allowed Rule* rule is applied to each field to ensure that the values in them are normalized to the proper localization. Then, the `DetailFix()` action is applied to calculate a single missing value on each line item.

## The Export task profile

The Export task profile consists of five rulesets that are ready for immediate use. However, it does not contain rulesets to export the data to your imaging system or to your business application system. Because these rulesets are highly variable, the demo writes the data to an XML file. You must add additional rulesets to this task profile for a production installation.

Not all of the rulesets in this task profile are used to export data. Many of them prepare the data for export. Others handle the disposition of problem documents, such as those documents that must be reviewed or rescanned.

The Export task profile includes the rulesets (Figure 36):

- ▶ The SetStatuses ruleset
- ▶ The PreExport ruleset
- ▶ The Export ruleset
- ▶ The ExportClose ruleset
- ▶ The RoutingNotification ruleset

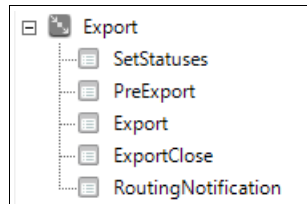


Figure 36 Export task profile

### The SetStatuses ruleset

There are two methods for marking documents for deletion, rescan, and review. The Datacap Desktop Verify task does this through user keystrokes and sets statuses on the documents and the pages for you. The thin client verify panels rely on a drop-down list in the *Routing\_Instructions* field to mark documents. The SetStatuses ruleset (Figure 37 on page 35) consolidates the two methods. This way, at export time, you only have to check the statuses or the *Routing\_Instructions* field to determine whether you want to export them or send a notification to someone.

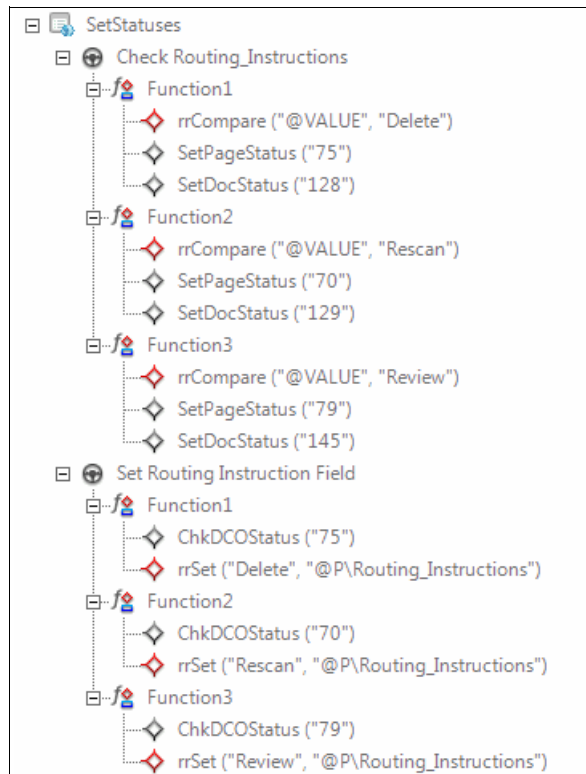


Figure 37 The SetStatuses ruleset

The first rule, Check Routing\_Instructions, runs on the Routing\_Instructions field. It looks at any value other than the default value of *None* and sets the statuses on the page and document accordingly.

The second rule runs at page level and checks the current page status that a thick verify client might have set. If the current page status is set, the rulesets the Routing\_Instructions field to the appropriate value.

Because of this ruleset, you only need to check one of the methods for marking documents for special handling with the rest of the Export ruleset.

## The PreExport ruleset

The PreExport ruleset is a catch-all for everything that needs to be done before exporting the documents and images. It is also where you can find *Intellocate*, one of the most important features of learning applications.

Intellocate is the technique that allows Datacap applications to “learn”. Documents that have never before been processed by the system are added to the fingerprint library. Locate rules are then used to attempt to find some of the data from these documents by using keyword searches or regular expressions. However, data that cannot be found automatically can be identified and captured quickly and easily by a verify operator by using the Click N Key capability.

After this task is done, Intellocate saves the zones for the fingerprint. Then, the next time a similar document is encountered, the fingerprint is matched, and all of the data is read by the zones.

Figure 38 on page 36 shows the non-Intellocate rules that are associated with the PreExport ruleset.

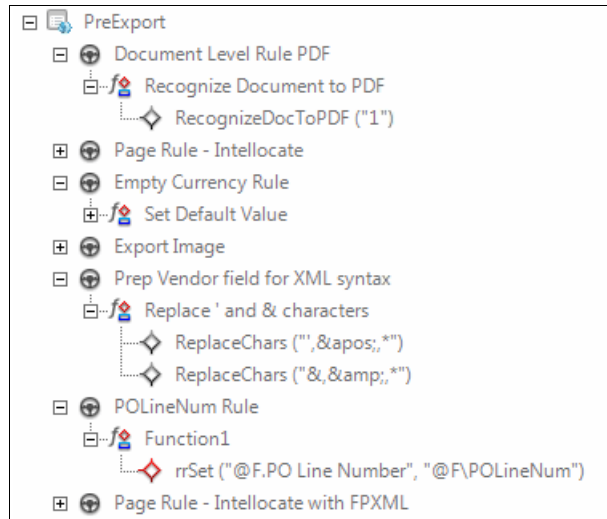


Figure 38 Non-Intellocate rules in the PreExport ruleset

The document-level rule uses the Nuance recognition engine to make a text searchable PDF. This PDF is stored in the batch directory and is named with the DocumentID property and a .pdf extension.

The empty currency rule attaches to each field that contains currency and defaults the value to 0.00 if the field is blank. This rule might need to be changed to 0,00 if a comma is used as a decimal value in the system to which you are exporting.

The Prep Vendor Field for XML Syntax rule checks the vendor field for an apostrophe (') or an ampersand (&) and replaces those characters with the XML equivalents to those characters. The PO LineNum rule ensures that the PO LineNumber field is populated with data from a POLR lookup.

The Intellocate ruleset makes up the bulk of the processing in the PreExport task profile. Figure 39 on page 37 shows the first part of the ruleset, Page Rule - Intellocate rule. The rule runs on the Main\_Page object.

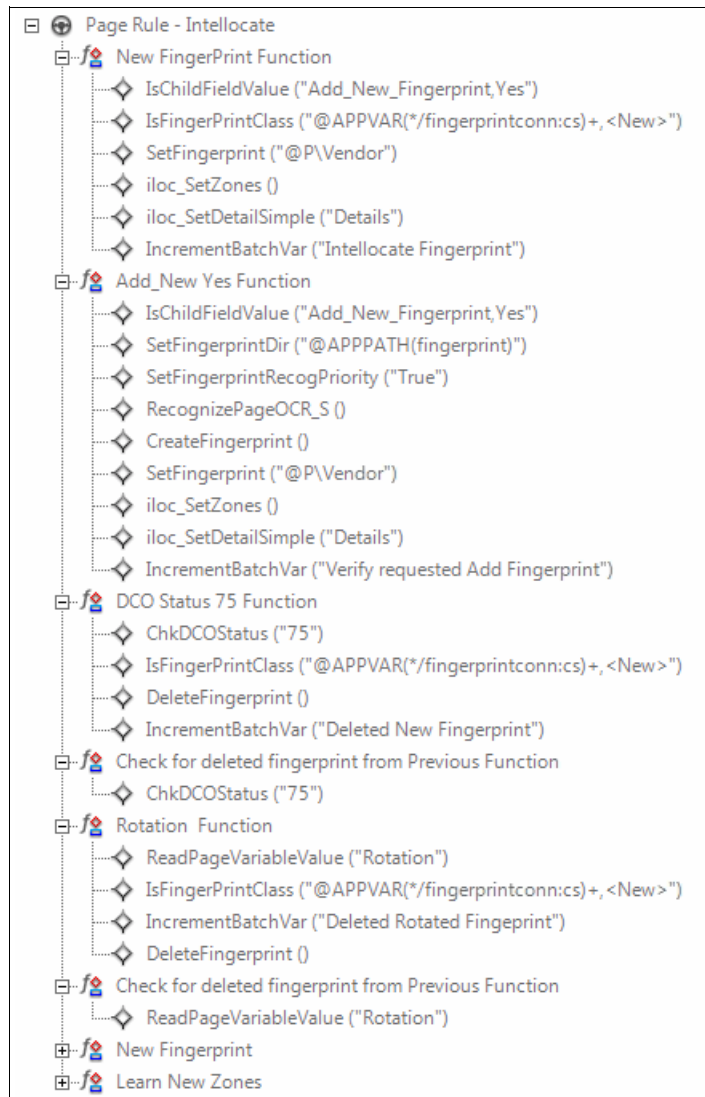


Figure 39 First part of the Intellocate ruleset

The first two functions control what happens if a data entry operator adds a new fingerprint based on the current image. Normally this is only done if there is a fingerprint mismatch. However, sometimes data entry operators mistakenly add a fingerprint when the fingerprint is already in the <New> classification, which is created anyway.

For the first function to complete successfully, the operator must choose NewFingerprint when it is presented to them. The fingerprint is in <New> to correct a misunderstanding that some operators have. If the fingerprint is in <New>, Intellocate saves the zones for the fingerprint. We do not want to create an additional new fingerprint. If this happens, the SetFingerprint action classifies the fingerprint (moves it out of the <New> classification) into a classification that matches the Vendor name. The iloc\_SetZones sets the header fields, and the iloc\_SetDetailSimple sets the zones of the detail fields.

**Important:** These actions write the zones to the Setup DCO. If you want to use FPXML, unhook this rule from the page and replace it with the unattached rule in this ruleset for FPXML.

In its essence, Intellocate is done with three actions: SetFingerprint, which moves the fingerprint out of <New> and classifies it, and the two iloc actions that save the zones. The rest of these actions ensure that these actions need to be done.

The second function runs if the data entry operator clicked the **NewFingerprint** button correctly this time because the document matched an existing fingerprint erroneously. When this happens, we want to dynamically create a fingerprint in the library from the existing image, run Intellocate on it to classify it, and save the zones.

Because the page might have a multi-CCO (MCCO), we re-recognize the page to ensure that it creates a single page CCO for the first page. We also indicate where to store the new fingerprint that it creates. We use CreateFingerprint to create the fingerprint before we use Intellocate.

The DCO Status 75 function handles deleted documents. If an operator indicates that a document does not belong in the system, we want to delete any fingerprint that it created. This function can return a value of *false* if the fingerprint was somehow already deleted. Therefore, the trailing function checks this value and returns *true* so that the rule does not continue.

The same process is done for rotated documents. You do not want to store inverted fingerprints. Therefore, this function deletes them.

If a fingerprint remains in <New> after these functions have checked for special processing, the New Fingerprint function runs (Figure 40).

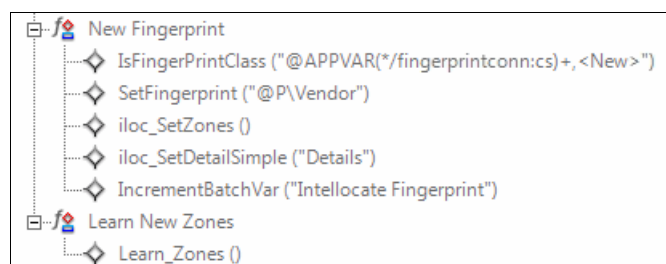


Figure 40 Remaining part of the Intellocate rule

As you can see, this process takes any fingerprint that is still in <New>, automatically classifies it, and saves the zones.

If a fingerprint is not in <New>, the LearnZones action examines the PreVerify Position variable that was created in the Routing ruleset and the current position of the field in the runtime DCO. If the PreVerify Position was 0,0,0,0 and the operator provided a zone for the data, the position of that field is added to the other fingerprint positions.

## The Export ruleset

When you demo Accounts Payable Capture, you might not have access to a business application system or an imaging system. Therefore, for demo purposes, we write out a standard text file with XML tags. Because this ruleset is not commonly used for production, it is not explained here. This file writes the text searchable PDF documents and data in XML format in the APT\Export directory.

## The ExportClose ruleset

The ExportClose ruleset (Figure 41 on page 39) writes the XML tags to complete the XML file and closes the file. This ruleset is not used in production.

However, you might want to add the SetFPStats() action in the ExportClose ruleset to your own custom export. This action returns the number of times, as well as the most recent time, that a particular fingerprint was used. This information helps you manage the fingerprint library, so you can remove fingerprints that are no longer in use.

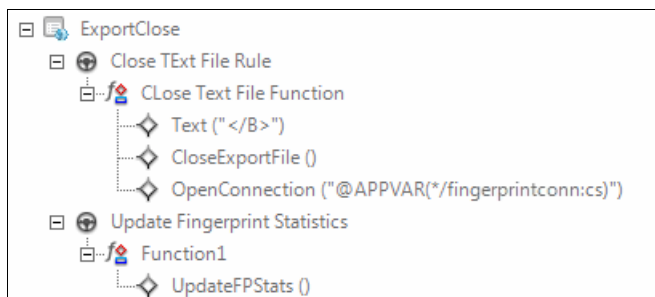


Figure 41 UpdateFPStats action that must be in your export

## The RoutingNotification ruleset

The RoutingNotification ruleset is also altered for production. This ruleset must be in place if you want to notify someone that the batch had documents in it that could not be processed (Deleted, Rescan, or Review).

Figure 42 shows the RoutingNotification ruleset.

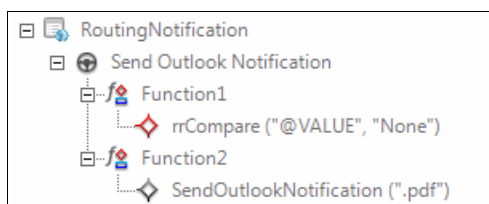


Figure 42 RoutingNotification ruleset

The first function checks the Routing\_Instructions field. If it is not set to *None*, it fails. The failed document is then processed by SendOutlookNotification, which sends an email to a person specified in the settings.ini file and attaches the multipage PDF file.

The reason this action is not used in production is that it requires Microsoft Outlook to be set up on the background machines. The version of Microsoft Outlook must support sending email messages programmatically, but not all versions do. You might need to replace this action with another email action to send the documents. Alternatively, develop another method of informing someone of problem documents, for example using an entry in a database that is polled or, which triggers an alert of some sort.

## Authors

This paper was produced by a specialist working at the International Technical Support Organization, San Jose Center.

**Jan den Hartog** is an Enterprise Content Management Technical Sales Specialist in the United States. Jan has over 18 years experience in product management, technical pre-sales, system integration, single-source publishing, and document transformation and migration. His experience, together with time spent in a support role, has taught him how to explain complex technical concepts in a clear and understandable manner.

**Tom Stuart** is the IBM Executive Enterprise Content Manager Capture Alchemist and is part of the IBM North America Software Advanced Technology (SWAT) team. He is responsible for developing easy-to-replicate techniques that help customers rapidly deploy complex data capture applications, most of which are integrated with other business applications. Tom is also closely involved with refining prototype solutions and has the application developer responsible for IBM Datacap Taskmaster Accounts Payable Capture and the IBM Datacap product Taskmaster Flex. He previously directed the pre-sales engineering activities of Datacap. Tom has a dual background in education and product development. He is widely quoted on technical issues and frequently addresses industry conferences.

Thanks to the following people for their contributions to this project:

Whei-Jen Chen  
LindaMay Patterson  
International Technical Support Organization, San Jose Center

## Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Stay connected to IBM Redbooks

- ▶ Find us on Facebook:  
<http://www.facebook.com/IBMRedbooks>
- ▶ Follow us on Twitter:  
<https://twitter.com/ibmredbooks>
- ▶ Look for us on LinkedIn:  
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new IBM Redbooks® publications, residencies, and workshops with the IBM Redbooks weekly newsletter:  
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:  
<http://www.redbooks.ibm.com/rss.html>



# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

© Copyright International Business Machines Corporation 2015. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

This document REDP-5235-00 was created or updated on July 17, 2015.


Send us your comments in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:  
[ibm.com/redbooks](http://ibm.com/redbooks)
- ▶ Send your comments in an email to:  
[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)
- ▶ Mail your comments to:  
IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400 U.S.A.

## Trademarks

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 US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Redbooks (logo) ®  
IBM®

Redbooks®  
Redpaper™

The following terms are trademarks of other companies:

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

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





REDP-5235-00

ISBN 0738454311

Printed in U.S.A.

Get connected

