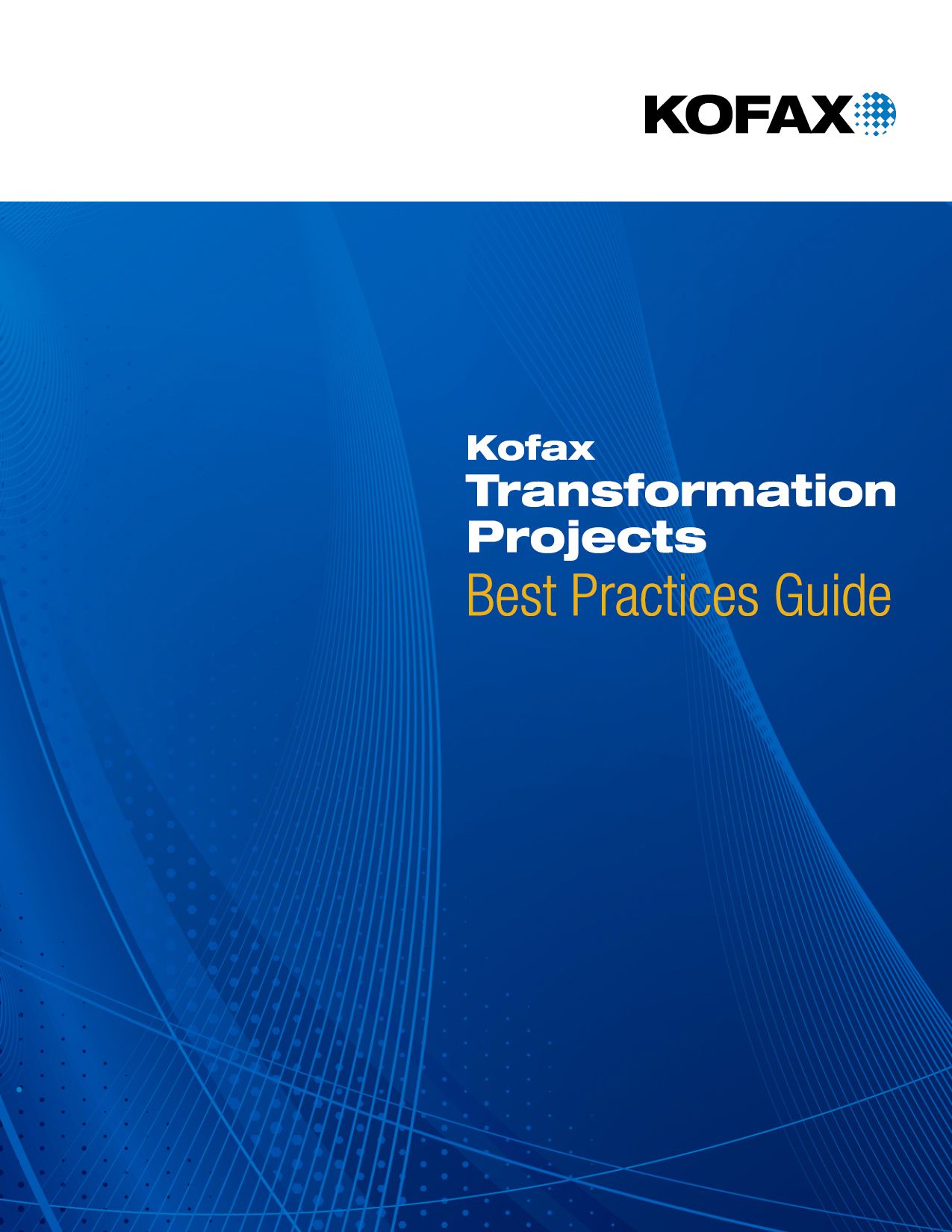
Field.Enablement@Kofax.com 

Field.Enablement@Kofax.com

Version 1.0 / May 2014

# Logo_KOFAX_RGBContents

To jump to a specific section, click its title here:

[Contents 2](#_Toc386812489)

[Who Should Read This Book? 6](#_Toc386812490)

[Cautions and Disclaimers 6](#_Toc386812491)

[Version History 6](#_Toc386812492)

[Introduction 6](#_Toc386812493)

[Highlights 7](#_Toc386812494)

[Principles of Success in KTM 8](#_Toc386812495)

[The Goal of every KTM Project is Human Productivity 8](#_Toc386812496)

[The Goal of a KTM project is Not Accuracy 8](#_Toc386812497)

[1. Accuracy does not solve the problem of the user. 9](#_Toc386812498)

[2. Accuracy metrics cannot be converted into ROI. 9](#_Toc386812499)

[3. Everyone will be focusing on the wrong goals. 9](#_Toc386812500)

[4. Kofax builds Smart Process Applications 9](#_Toc386812501)

[Focus on Kofax Core Business – helping people communicate 10](#_Toc386812502)

[KTM’s job is to understand data and make it actionable 10](#_Toc386812503)

[KTM is not like any software you know. 11](#_Toc386812504)

[Business Analysis 12](#_Toc386812505)

[Customer Qualification 12](#_Toc386812506)

[Some Examples 14](#_Toc386812507)

[Customer Walkthrough 15](#_Toc386812508)

[Document Analysis 15](#_Toc386812509)

[Document Analysis Workshop 15](#_Toc386812510)

[Representative Document Collection 16](#_Toc386812511)

[Data Collection 17](#_Toc386812512)

[Key Value Pairs 18](#_Toc386812513)

[Field Analysis Workshop 18](#_Toc386812514)

[Goals of the Field Analysis Workshop 18](#_Toc386812515)

[People you should invite to the Field Analysis Workshop 18](#_Toc386812516)

[Which fields should KTM deliver? 19](#_Toc386812517)

[Validation Fields 20](#_Toc386812518)

[Agreeing on your Data 20](#_Toc386812519)

[Writing your Statement of Work 21](#_Toc386812520)

[Metrics 21](#_Toc386812521)

[Accuracy Metrics 21](#_Toc386812522)

[1. Character Level Confidence 21](#_Toc386812523)

[2. Character Level Accuracy 21](#_Toc386812524)

[3. Field Level Accuracy 22](#_Toc386812525)

[4. Precision/Recall 22](#_Toc386812526)

[5. True/False Positives/Negatives 22](#_Toc386812527)

[Performance Metrics 23](#_Toc386812528)

[6. People per Project 23](#_Toc386812529)

[7. This number is easily convertible to ROI.Documents per Person per Hour 23](#_Toc386812530)

[The Priorities of a Transformation Project 23](#_Toc386812531)

[Benchmarking 23](#_Toc386812532)

[Value of Benchmarking 24](#_Toc386812533)

[Benchmarking Table Data 25](#_Toc386812534)

[Training Data and Test Data 26](#_Toc386812535)

[How many Training Documents should I have? 26](#_Toc386812536)

[How to fix mistakes found by a KTM Extraction Benchmark 27](#_Toc386812537)

[#1 Remove False Positives 27](#_Toc386812538)

[#2 Reduce False Negatives 28](#_Toc386812539)

[#3 Minimize True Negatives 29](#_Toc386812540)

[Further Interpretation of a Benchmark 31](#_Toc386812541)

[Creating Golden XDocuments 32](#_Toc386812542)

[Implementing a KTM Project 33](#_Toc386812543)

[Build a Project that learns 33](#_Toc386812544)

[Steps in Building a KTM Project 34](#_Toc386812545)

[Clear Separation of Logic and Data 35](#_Toc386812546)

[Why you should avoid the Format Locator as much as possible 35](#_Toc386812547)

[User Experience 36](#_Toc386812548)

[Steps to building a great User Experience – Fingers , Eyes & Mind. 36](#_Toc386812549)

[User Productivity 36](#_Toc386812550)

[Fingers 36](#_Toc386812551)

[Eyes 39](#_Toc386812552)

[Mind 41](#_Toc386812553)

[Classification 43](#_Toc386812554)

[Do you really need to classify documents? 43](#_Toc386812555)

[Classification Trees and Field Inheritance 44](#_Toc386812556)

[“Structural” Classes 45](#_Toc386812557)

[Building Training Sets 46](#_Toc386812558)

[Classification Tools 46](#_Toc386812559)

[1. Extracting automatically from a Document Management System. 46](#_Toc386812560)

[2. Windows Explorer 46](#_Toc386812561)

[3. Build a KTM project simply for Classification 47](#_Toc386812562)

[4. Document Clustering 47](#_Toc386812563)

[5. Trainable Document Separation (TDS) 48](#_Toc386812564)

[6. Project Builder 48](#_Toc386812565)

[Classification Methods 48](#_Toc386812566)

[1. Layout Classification 48](#_Toc386812567)

[2. Text Classification 48](#_Toc386812568)

[3. Instructional Classification 48](#_Toc386812569)

[4. Script Classification 49](#_Toc386812570)

[1. Fuzzy Database Queries from Script 50](#_Toc386812571)

[5. Classification Locator 51](#_Toc386812572)

[6. Manual Classification (Document Review & Validation) 51](#_Toc386812573)

[7. Trainable Document Separation 51](#_Toc386812574)

[Classification Tips 51](#_Toc386812575)

[How many documents should I train for Classification? 51](#_Toc386812576)

[Customer Classification is not always KTM Classification 52](#_Toc386812577)

[Divide and Conquer 52](#_Toc386812578)

[Find and Filter 52](#_Toc386812579)

[Document Separation 52](#_Toc386812580)

[1. Use Barcode Stickers 52](#_Toc386812581)

[2. Scan documents individually 53](#_Toc386812582)

[3. Use Separator Sheets 53](#_Toc386812583)

[4. Combine all three methods 53](#_Toc386812584)

[5. Rubber Stamp with Datamatrix barcodes. 53](#_Toc386812585)

[6. Trainable Document Separation (TDS) 53](#_Toc386812586)

[7. Automatic Document Separation for Invoices 53](#_Toc386812587)

[Locators 54](#_Toc386812588)

[Training 57](#_Toc386812589)

[How to Train Documents in Project Builder 57](#_Toc386812590)

[Selecting Anchors 58](#_Toc386812591)

[How to Select Anchors 58](#_Toc386812592)

[Trainable Locator Subfield Confidences 59](#_Toc386812593)

[Online Training Scoring and automatic conflict cleaning 59](#_Toc386812594)

[Resolving Conflicts 60](#_Toc386812595)

[Invoice Evaluator/Advanced Evaluator 60](#_Toc386812596)

[Create a strong Learning Feedback Mechanism for the User 60](#_Toc386812597)

[Databases 62](#_Toc386812598)

[Database Locator 62](#_Toc386812599)

[Vendor Locator 62](#_Toc386812600)

[Database Evaluator 63](#_Toc386812601)

[Spell Checking with Database Evaluator 63](#_Toc386812602)

[“Webservice” or “SQL” query with Database Evaluator 64](#_Toc386812603)

[Validation Form Search 64](#_Toc386812604)

[Search And Matching Server 65](#_Toc386812605)

[Kofax Search & Matching Server KSMS has the following value 65](#_Toc386812606)

[Comparison between different Database Types 65](#_Toc386812607)

[Tables 67](#_Toc386812608)

[Automatic Table Locators 67](#_Toc386812609)

[Table Model 68](#_Toc386812610)

[Table Header Pack 69](#_Toc386812611)

[Row and Column Detection 69](#_Toc386812612)

[Amount-based Algorithm 69](#_Toc386812613)

[Position-based Algorithm 70](#_Toc386812614)

[Header-based Algorithm 70](#_Toc386812615)

[Line-based Algorithm 70](#_Toc386812616)

[Layout-based Algorithm 70](#_Toc386812617)

[Manual Table Locator 70](#_Toc386812618)

[Interleaved Columns 71](#_Toc386812619)

[Embedded Columns 71](#_Toc386812620)

[Line Item Matching Locator 72](#_Toc386812621)

[Combining Locators 73](#_Toc386812622)

[Table Interpolation in Validation 74](#_Toc386812623)

[Localization 74](#_Toc386812624)

[Dictionaries 78](#_Toc386812625)

[Dates 79](#_Toc386812626)

[Batch Restructuring 81](#_Toc386812627)

[Workflow Agents not needed 81](#_Toc386812628)

[Splitting Batches 82](#_Toc386812629)

[Auditing a KTM Project in production 82](#_Toc386812630)

[Scripting Guidelines 83](#_Toc386812631)

[Productivity, before Accuracy 83](#_Toc386812632)

[Divide & Conquer 83](#_Toc386812633)

[Debugging 83](#_Toc386812634)

[Error handling 83](#_Toc386812635)

[Separate Data from Logic. 84](#_Toc386812636)

[Always assume data is bad, wrong or missing 84](#_Toc386812637)

[Put scripts in the right events 84](#_Toc386812638)

[Use well defined function names 85](#_Toc386812639)

[Refactor continuously. 85](#_Toc386812640)

[Use KTM‘s Object Model. This will simplify your code and make it easier to read. Avoid using Strings. 85](#_Toc386812641)

[How to write a Script Locator with a single Subfield 86](#_Toc386812642)

[How to write a Script Locator with multiple Subfields 86](#_Toc386812643)

[“In-Place” Locator Alternative Editing 88](#_Toc386812644)

[Multifield Script Locators 88](#_Toc386812645)

[Manipulating Trainable Locator Results 88](#_Toc386812646)

[Scripting Tables 88](#_Toc386812647)

[A word about .Net and scripting 89](#_Toc386812648)

[Useful COM Objects & Win32 Functions 89](#_Toc386812649)

[Barcodes 90](#_Toc386812650)

[Unicode 90](#_Toc386812651)

[Databases & Dictionaries 90](#_Toc386812652)

[Scripting Unicode 92](#_Toc386812653)

[Corrupted File Names in Windows 92](#_Toc386812654)

[Summary 93](#_Toc386812655)

[Appendices 93](#_Toc386812656)

[1. Document Collection With Kofax Express 93](#_Toc386812657)

[2. Useful KTM Objects and Properties 95](#_Toc386812658)

[3. Visual Basic 6 tips 98](#_Toc386812659)

[4. Fuzzy Database Queries from Script 99](#_Toc386812660)

[5. How to Classify by Barcode in Script 101](#_Toc386812661)

[6. How to Classify by Phrase in a Region on a Document 103](#_Toc386812662)

[7. SQL Queries from Script 104](#_Toc386812663)

[64 bit ODBC Connection 104](#_Toc386812664)

[Using CscSQLDataTable for SQL “SELECT” 104](#_Toc386812665)

[Using ActiveX Data Objects for SQL “INSERT” and others 104](#_Toc386812666)

[8. Calling a Fuzzy Database in a Field Formatter for Spell Checking 106](#_Toc386812667)

[9. Calling a Fuzzy Database in a Validation Rule 108](#_Toc386812668)

[10. Create Golden Files from a Productive System 109](#_Toc386812669)

[11. Create Golden Files from CSV Data 110](#_Toc386812670)

[12. Integrating Webservices & Kapow with KTM 112](#_Toc386812671)

[13. Converting Fields to Dates and Amounts 115](#_Toc386812672)

[Index 116](#_Toc386812673)

# Who Should Read This Book?

This book was written for sales engineers and implementation consultants at Kofax and its business partners. It will help them understand their customer, identify requirements and design and deliver great solutions based on Kofax Transformation Modules (KTM). The design principles offered in this book will help deliver a superior user experience when validating documents and yield savings in time, money and effort.

# Cautions and Disclaimers

This book is called a Best Practices Guide**.** A best practice is a method or a technique that has consistently shown superior results over other possible approaches. The suggestions and guidelines contained in this document should help you build a better KTM project but the nature of this technology means there are many other design options that are not detailed here. Readers are encouraged to explore all of the choices available to them when working on such projects.

In making this book available, neither Kofax nor the individuals who participated in its creation make any representation regarding its content and the value of the guidance offered. Each user of this document is solely responsible for the information contained within it and assumes all risks in connection with its use.

# Version History

Version 1.0, April 24, 2014. Launch Version

# Introduction

Many market-watchers are saying that we have now entered the age of the customer. Prior to this transition all of the power was in the hands of the seller. Recent technological advancements like the internet, mobile and social media have dramatically changed landscape. Consumers are now much more empowered and are able to make informed decisions on what they buy, when and from whom. Many companies find these new technologies very disruptive regarding their traditional way of doing business, and see themselves confronted with tough competition, fierce price pressure and an eroding customer base. Market leaders have understood that they need to offer more communication channels to their customers, improve their communication (both speed and relevance) and make business processes more transparent – in other words, improve their customer service and interaction.

You may ask: “Ok, but what has this to do with Kofax transformation projects?”. The response is “quite a lot”. The true mission of Kofax transformation is not related to bits and bytes, but to help improve communication, relieve employees from boring repetitive tasks and make them more productive, resulting in an improved customer experience. However, to unleash the real power of KTM, it is important to not only look at technical aspects, but follow an implementation methodology that focuses on productivity gains and happy users.

This book is:

* a best practices guide for conducting customer workshops and using Kofax Transformation Project Builder.
* a compilation of content you need to be familiar with before building a KT project – any advanced and optional content is in the Appendices.
* a living document that you can contribute to. Email changes or suggestions to [field.enablement@kofax.com](mailto:david.wright@kofax.com).
* the result of materials produced in numerous KTM workshops conducted in North and South America, Australia, Asia, and Europe throughout 2013 with nearly 200 participants from Kofax and Partners presales and professional services.

This book is not:

* a substitute for Kofax Transformation product training – it assumes you understand what locators, fields, classes, formatters, and validation rules are.
* a substitute for the KTM Wiki, which contains code snippets and techniques <http://knowledgebase.kofax.com/faqsearch/results.aspx?QAID=17911>
* a guide for installing and configuring Kofax Transformation Modules and Kofax Search and Matching Server. It does not refer to Kofax Capture or KCNS.

The content of this book applies to all four Kofax Transformation Products, despite the use of the term “KTM” throughout.

* Kofax Transformation Modules 5.5 and 6.x
* Kofax Transformation Toolkit
* Kofax Total Agility 7.x
* Kofax RealTime Transformation Interface (RTTI), which hosts a KT project in a webservice.

# Highlights

1. KTM projects are about improving human productivity, not automating tasks.
2. Analyze your customers’ corporate vision, organizational goals, and departmental needs and pains before anything else.
3. Conduct a Customer Walkthrough, following the paper, looking for “process crimes” to completely understand the context of your project.
4. Run a Document Analysis Workshop with your customers to teach them how to “do KTM” with pen and paper.
5. Your customers do their own Data Collection to build a Representative Document Set and an Excel Sheet of field values.
6. Run a Field Analysis Workshop to agree on all document classes, fields, processes and data across the organization.
7. Write your Statement of work based on the representative documents and field data collected and human performance metrics.
8. Build your project’s classes, fields, user Interface, formatters and validation rules in Project Builder focusing on human productivity.
9. Build, test, and tune locators on benchmark sets, not on individual documents.
10. Teach your project with lots of examples and databases. Don’t program it with rules.
11. Install a high performance project at your customer on Day One and expect great results.

# Principles of Success in KTM

You want to have a successful KTM project with happy users, who love your solution because it helps them do their daily jobs. If your users can come to work in the morning, turn on their computers, see the Kofax Splash Screen and smile, then you have been successful.

**A true story**. A user had 24 clerks manually processing 800 invoices per day - each. They bought a KTM solution which was technically well planned, benchmarked, and built over three to four months. The clerks were now processing 1200 invoices per day and unhappy with only 50% improvement. After two weeks of further effort focusing on the user experience, the clerks were processing 2500 invoices per day each – an additional 100% improvement. Focusing on human productivity in KTM was ten times more effective than focusing on accuracy in KTM.

|  |  |  |  |
| --- | --- | --- | --- |
| **Stage** | **Productivity** | **Improvement** | **KTM Project Effort** |
| Manual data entry | 800 | - | - |
| KTM “Accuracy” solution | 1200 | 50% | 3 months |
| KTM “Productivity” solution | 2500 | 100% | 2 weeks |

## The Goal of every KTM Project is Human Productivity

The goal of every KTM project is to reduce costs of document processing by improving human productivity. The “enemy” or “competition” of a KTM project is people doing work manually.

Productivity in a KTM project is easy to measure.

* **Documents/Persons/Day**
  + This metric is used when the goal is to make people more productive.
  + A person may be able to process 800 documents a day manually, but KTM could enable them to process 2500 per day.
* **Persons/Project**
  + This is used when a team is trying to downsize or reallocate staff. The document processing might require 30 people, and they would like to require only 8-10 people.

Both of these metrics are easily and accurately measurable and convertible into money-saving and hence ROI. You do not even need any technical specifications to calculate such numbers.

If you always focus on the people in your project, you will be able to manage expectations better and it will be easier to sell. You will have more acceptance criteria that you can fulfill, and you’ll have better good will.

## The Goal of a KTM project is Not Accuracy

KTM projects that focus on accuracy have more problems and a higher risk of failure and user dissatisfaction.

### Accuracy does not solve the problem of the user.

There are many reasons why accuracy is a poor goal of a KTM project. It masks the real problem - users do not have an accuracy or speed problem. These can be fixed by adding more people to the team. Users have a people and money problem.

If you identify the real problem that needs to be solved, you can build a better KTM project that meets the real need of the user.

A customer may say, “We need to OCR documents,” when they actually mean, “We need to get accurate and timely data from our documents into our system of record.” Their problem is about accuracy and time, but the conversation is about OCR, which they think is their solution.

To find what is really needed, find out what is behind the customer’s technology request. If you focus on accuracy, you will focus on technology and not on people.

### Accuracy metrics cannot be converted into ROI.

There are many accuracy metrics – most are not helpful, and none convert into ROI. Here are some common accuracy metrics:

* Character accuracy
* Field accuracy
* Straight-through processing rate (document accuracy)
* Classification accuracy

None of these metrics can help you determine the human cost of the remaining documents, because you will always have to guess the human cost of correcting the errors.

Accuracy metrics don’t consider the difference between False Negatives, True Negatives and False Positives True/False Positives/Negatives.

### Everyone will be focusing on the wrong goals.

Accuracy metrics often come with arbitrary targets – and great effort is directed at improving these scores rather than making people productive. If the Statement of Work says 80% field accuracy and you have 76.5% field accuracy, should you be expending effort for that last 3.5%? Probably not. The effort to remove the last 3.5% probably produces diminishing returns, while effort spent in productivity gives far greater gains.

You will see later that accuracy is only priority #4 in a KTM project. Your users, however, often have strong accuracy expectations. The next section will prepare you to counter these arguments, and also educate your customers on valuable priorities and metrics that will benefit them in their KTM Solution

### Kofax builds Smart Process Applications

In today’s world, it’s not IT departments – it’s business units that focus on customer value - that drive software solutions. Our metrics need to focus on this as well. Productivity, user experience, and transparency are more important than metrics that are easy for IT to set and measure.

## Focus on Kofax Core Business – helping people communicate

Kofax is in the business of helping companies communicate with each other and their customers, regardless of the medium of communication (it could be paper, phone, fax, SMS, mobile app, email, social media….). KTM doesn’t care about the medium, and neither should you.

## KTM’s job is to understand data and make it actionable

A Customer, Fred, is sending a message to a company. KTM’s job is to:

* Read the message
* Understand the message
* Interpret/translate the message into the language of Person B so that they can act upon it.
  + *Fred says “I am Fred Smith”.* company *needs to hear “Customer=45693”*
  + *Fred says “29th October 2010” and* company *needs “Date=20101029”*
  + *Fred says “I am disappointed…” and* company *hears “Complaint=true”*
* “Fred Smith” is not actionable data. There may be one or more Fred Smiths or even “Frederick Smiths.” The customer number 45693 is actionable – and this is KTM’s job – make people’s work easier and help them find what they need to know to do a job.

Your job with KTM is to make it easier for Person B to understand and act on Person A’s information. Don’t think about your job as locators, databases, formatters or OCR – think of it always about decrypting and transforming a message.

## KTM is not like any software you know.

Most software is deterministic – certain input produces certain output 100% of the time. SQL, Web, JavaScript, DMS, ERP are all examples of deterministic software.

KTM is non-deterministicsoftware - you cannot be 100% sure which results will be produced.

Probabilistic learning software is critical to the success of software used in speech recognition, machine translation, robotics, genetics, astronomy, weather, earthquakes and many other areas. It is increasingly being used in business software – “Big Data,” “Business Analytics”.

In later chapters we will look at how to build your KTM project non-deterministically.

#### Definition of Deterministic and Probabilistic Software

|  |  |
| --- | --- |
| Deterministic Software | Learning Software |
| Definition. Software that produces predictable results using rules and logic. | **Definition**. Software that produces probabilistic results using training data. |
| SQL, HTML, javascript, DMS, ERP, Yahoo. Most software until 2010 | OCR, Speech Recognition, Diagnostic Systems, VRS, KTM, Google Search/Translate/Car, Analytics, Big Data, Genetics, AI, Neural Networks. Software after 2010. |
| Logical, predictable, “easy” to fix bugs, programmable, rigid, “slow” | Learning, big-data, knowledgeable, statistical, probabilistic, flexible, “fast” |
| Many rules | Few or no rules |
| Tends to be complicated | Tends to be **complex**, and can allow **emergence** (<http://en.wikipedia.org/wiki/Complexity#Study_of_complexity> ) |
| Programmable with rules. *convert human behavior to a set of rigid rules* | Trainable with data, lots of data. *mimic human behavior.* |
| High development effort. | Low development effort. |
| You studied this at university before 2010 | You studied this at university after 2010 |
| Handles clean input well. | Handles messy input well (such as OCR, written language)  Always performs better than deterministic software on messy data. |
| Developers need to improve it. Cannot learn. | Can improve itself with user feedback. |
| Cannot cover all scenarios. Will eventually fail. [Gödel’s Incompleteness Theorem](http://en.wikipedia.org/wiki/G%C3%B6del%27s_incompleteness_theorems) *“A rules-based system cannot demonstrate its own consistency”* | Not trying to be perfect. Trying to be useful. Like a human. |
| Focus on perfection, no compromise. | Focus on productivity and compromise. *Google Search is not a failure if what you are looking for is occasionally on page two of the search results* |
| Acceptance criteria are based on perfect technical performance | Acceptance criteria are based on usefulness and productivity |

Software for language translation, speech recognition, Search, Natural Language Processing progressively abandoned deterministic approaches in the 1980s and 1990s for training data-based approach.

The advantages for you with your KTM project

* You don’t need to spend time configuring lots of rules and locators in KTM.
* Your users can be more involved in building your project – you can share the burden.
* If you use the learning systems and data from your users, you can build high quality, accurate, transparent, fast and manageable projects.

The following sections will teach you how to use probabilistic techniques for project success.

# Business Analysis

Here we will go through the six steps required to completely understand a customer’s requirements and collect everything you need to build a KTM project.

1. **Customer Research Qualification** ‒ understand your customer completely.
2. **Customer Walkthrough** ‒ meet the people, understand their processes, and analyze the “crime scene” of their workplace.
3. **Document Analysis** ‒ help your customers understand their documents completely.
4. **Representative Document Collection –** your customer provides all documents needed for the project.
5. **Data Collection** ‒ your customer collects all of the data needed for the project, mostly from databases.
6. **Field Analysis** ‒ understand together with your customer what the KTM project should do.

Before every KTM project, you should have thoroughly completed these steps before starting to build the project. That means you have effectively “completed” the proof of concept using paper and Microsoft Excel before you actually start to build the real Proof of Concept in Project Builder.

If you follow these three steps you will have a project that fits your customers’ requirements, your customer will be happy and engaged in the KTM solution, and you will have a straightforward implementation with excellent productivity from Day One.

## Customer Qualification[[1]](#footnote-2)

This chart is a summary of more than 100 KTM projects throughout the world in 2013. Use it to discover the vision, goals, needs, and pains of your customer. It also helps you to understand your customer, prioritize project goals, and communicate to different stakeholders in your project.

Do this research before you visit the customer and take it with you to lead your conversation with the customer. Use this table as a checklist of topics to talk about with your customer to uncover needs and pains you might have missed.

|  |  |  |
| --- | --- | --- |
| Corporate Level | Focus | Explanation and checklist. |
| Executive | **Vision** | *why the company exists, purpose, mission* |
| Simplify, best in field, best services, innovate  What is the corporate vision on the website? | |
| Operations | **Goals** | *what the company is trying to achieve* |
| Improving speed, quality, service, growth, efficiency, market position, trust  “Automate”, transparency, customer satisfaction, prioritize  Save money, create value | |
| Department | **Needs** | *what they need to do in order to achieve a goal* |
| Comply, digitize, migrate, simplify, integrate, secure, retrieve, track, streamline, achieve “touchless”, measure  Keep employees happy and content, failure is not an option | |
| Department | **Pains** | *prevents them from reaching a goal* |
|  | Slow, not transparent, lost, wrong, complaints, increasing costs, complex, wasteful, no resources  Ask the discovery question, “What happens if we do nothing at all?” | |
| Product | **Features** | *What can the software do* |
| Confidence, Fields, OCR, Scanning, VRS, Databases, | |

Knowledge of the customer’s needs and pains is critical to the success of your project, and not just for the sales team. Every person configuring, coding, and working on a KTM project needs to consider the needs of the customer at all times to guide communication and good decision-making.

Goalsare *always* business-oriented and have nothing to do with technology. Here are some possible goals:

* Double the productivity of the finance organization within one year to support the expansion of our business.
* Consolidate all processes and tools across all acquired companies within two years to unify the company, save money, and to increase efficiency, reporting, and transparency.

These are not goals (they are either needs or product features):

* Automate processes.
* Achieve 85% OCR accuracy on scanned documents. *There is a false belief here that this will produce certain productivity targets.*
* Scan all incoming documents.
* Avoid losing documents

A need leads to a goal. A pain prevents a goal.

Always ask the paindiscovery question:

*“What happens if we do nothing at all?”* These are needs:

* Digitize document processing to reduce archive costs, and speed up processing across many locations.
* Process documents faster so we can gain early payment discounts.

These are pains:

* 20% of documents are lost or take more than five hours to find after one year.
* Many customers call us complaining about how slow we are.
* 15% of receipts we pay are for the wrong amount, and we are, on average, 23 days late in payment.

Create a vision/goal/needs/pain chart for your customer. With your customer, prioritize their goals, needs, and pains. Send the chart to your customer for written confirmation that everyone understands the customer scenario.

You can now use this chart to communicate better about your project. You should not talk with corporate executives about product features or departmental pains directly, but start at their vision language and level, and diving down to the operation goals level to communicate. Communicate to each level at their level, referring either to one level up or down, and occasionally going to further levels. If you do this well, it will help you communicate the value of your KTM project more effectively.

This chart will also help you focus on creating a KTM project that exactly matches what your customer needs.

### Some Examples

|  |  |
| --- | --- |
| A government department | |
| Executive Vision | Be a model state agency, delivering exceptional customer service and promoting effective and efficient business processes, professional employees, innovative technology and strategic partnerships |
| Operational Goals | Administer the State's licensing and titling laws: • maintain strict controls to deliver secure and valid identification, licenses, property records • accurately account for the receipt and timely distribution of all revenue collected • provide the best customer service to serve our citizens |
| Departmental Needs | • Reduce the time of the front counter experience for customers • Improve customer service response time • Implement and administer Legislative mandates for Records Retention policies • Provide platform for long-term scalable and sustainable growth |
| Departmental Pains | Slow, not transparent, lost, wrong, complaints, increasing costs, complex, wasteful, no resources |

|  |  |
| --- | --- |
| Land Registry Office | |
| Executive Vision | To make property transactions secure, integrated, accessible, and easier for all.  To become an efficiently operated and successful institution that guarantees geo-referenced real property related rights for facilitating secure property transactions and providing an infrastructure and services for economic, environmental, and social purposes. |
| Operations Goals | Create a profitable value-added service by selling data for research, analysis and statistics. |
| Department Needs | Turn existing paper document archives into valuable, structured data. |
| Department Pains | Not able to use available funding. Not able to deliver. |

## Customer Walkthrough

Perform a customer walkthrough early in the sales cycle. Walk around the department at your customer’s facility, observing exactly what happens with their documents and processes. Interview people about what they are doing and why they are doing it. Take photos if possible. Act like a detective on a crime scene looking for clues and evidence of document and process crimes. This walkthrough will help you understand where the customer’s needs are. You will also gain new ideas and experience that will help you with other projects.

This also helps you and your customer focus on the people and their processes rather than on the technology.

## Document Analysis

In this step, you will analyze the documents and their processes with your customer.

A customer may provide you with 5-10 samples (perhaps photocopies) and a list of required fields and expect you to build a solution. This is **not** enough to build a project.

Here are some key points for document analysis.

* Your customers knows their own documents. Make sure you interact with document experts and not IT project managers, who do not know the documents intimately.
* It is not your job to become a document expert. Your job is to transfer human knowledge about these documents to your KTM project. Your KTM project needs to mimic the humans, and the best way to do that is to learn lots of examples, not to build a rules engine.
* It is not your mission to build a magic black box that performs the people’s jobs better than they do, but to build a system that makes the people far more productive.

### Document Analysis Workshop

Hold a meeting with your customer for three to five hours. Make sure that all the relevant business personnel are there. The customer representatives should bring many photocopied samples of each document class. You will need lots of colored highlighter pens.

Teach your customer what classification, separation, and extraction are by using piles of paper and colored pens. Ask your customer to classify and separate documents and extract data. Ask them to explain how they made their choices. Do this for tens or hundreds of pieces of paper, until you all agree that everyone understands how the whole process will work for all of their documents.

If two people “extract” the same document differently, then your customer still does not know what data needs to be extracted.

It is important that the customer understand exactly what KTM does – nothing magical, but exactly what they have just done with pen and paper. Here is a picture of what KTM does – exactly what people do:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C:\Documents and Settings\SWilcox\Desktop\0 Project - BRAND\ALL_ICONS\Invoice.png | 🡪 | |  |  | | --- | --- | | **Field** | Value | | CLASS | Invoice | | Invoice Number | 345098 | | Invoice Date | 04.05.2013 | | VendorID | 56908 | |

While the customer is building this Excel sheet, they are also collecting together all their internal domain knowledge, data sources, and representative documents.

### Representative Document Collection

Your customer needs to provide you a collection of representative documents for the project.

“Representative documents”are good examples of the documents that the business sees from day to day.

* Exactly the same as the documents in the production system (size, kind, quality) 300 dpi TIFF group4 images, scanned with VRS, no marks or stamps[[2]](#footnote-3).
* You need at least a few days’ worth of documents. If your project is expected to process 20,000 documents per day, then you can expect your customer to provide you with 60,000 representative documents.
* Exactly the types of documents that are processed daily – a representative mix of good and bad/easy and difficult/readable and unreadable/printed/dot-matrix/handwritten/faxes/emails/common and rare and strange.
  + If, for example, there are special invoices with two tax rates or special discounts, they need to be in the representative set.
* The documents need to represent all processes and document classes that will be in the production system.
* These documents will define the entire scope and acceptance requirements of the project.
* Representative documents remove unwanted surprises.
* Representative documents form the benchmark sets you will use.
* When your customer understands what “representative” means, they will realize that 10-50 examples are simply not enough. Try to get at least 20 documents of each type, preferably 50 or more.
* A customer who says that they cannot provide you with 1000 examples, but expects your solution to process 1000 per day, is being neither realistic nor cooperative.

Agree with your customer in writing before project begin that these documents represent the quality, volume, classes, processes, and variety that will be in the production environment and that performance metrics measured on these documents are the acceptance criteria for this project.

### Data Collection

It is now the job of your customer to fill an Excel Sheet with field data from tens of documents. If the customer is unable to fill Excel with well formatted, consistent data from their own documents, they still have not understood the process and are not ready to build a KTM project. When this Excel sheet has been built, you know they have understood, mastered, and solved all of their data entry problems.

|  |  |  |
| --- | --- | --- |
| Image | **🡪** | Key-Value Pairs entered into Excel. |

Note the following

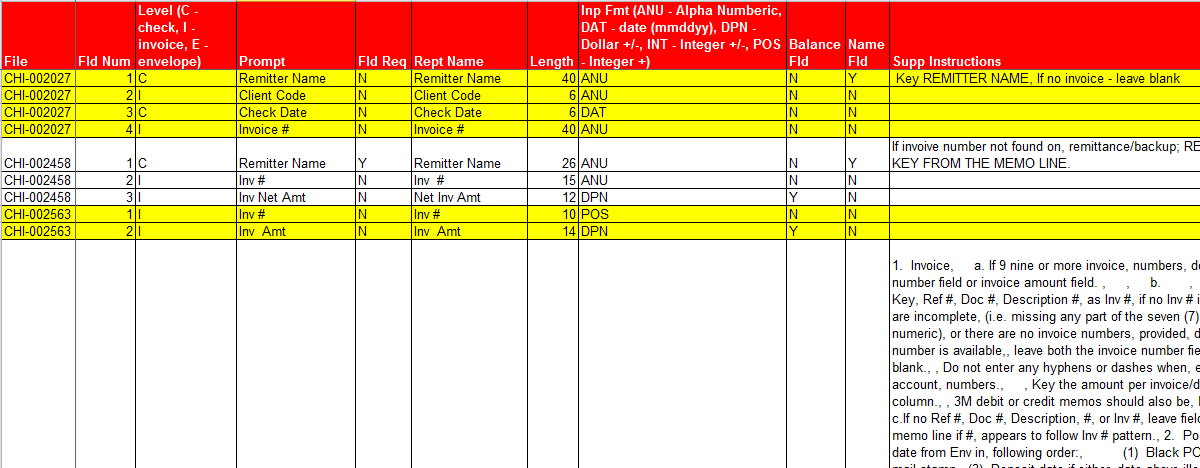
* Numeric and percentage fields are well formatted.
* PO Number is a table field (Use Alt-Enter to enter multiple rows in an Excel cell)
* Dates are formatted with dd/mm/yyyy, and not d/m/yyyy
* Vendor 1036 has no invoice date on the document, so the ScanDate should be used instead.
* 6.tif has a rounding error of 0.01 £ in the total.

### Key Value Pairs

Many customers will expect that it is enough to provide you with a list of document classes and required fields. Often a lot of effort has gone into the document – people have made an effort to write down rules about fields – but it generally is lacking. It is much better to obtain lots of real documents and analyze real field data.

There are usually complex descriptions of what field formats and validation should be right. This data is rarely completely reliable.

This data is deterministic data, and is not very useful in your non-deterministic project.



You need to take further steps after receiving such a document.

## Field Analysis Workshop

The Field Analysis Workshop is an important meeting that defines the entire scope and expectations of the KTM project.

If this meeting goes well, a simple invoice project could even go live after two weeks of effort, with a happy customer.

### Goals of the Field Analysis Workshop

* The customer completely understands the solution and its consequences internally
* The representative set of documents covers all customer requirements
* Number of fields that KTM extracts are minimized
* Databases provided to KTM from the business systems are maximized
* The project reduces effort for the maximum number of people, not just scan and validation operators, but process operators as well.
* The class hierarchy becomes clear and simple. You know which classes to define fields at, and they are inherited into their subclasses.

### People you should invite to the Field Analysis Workshop

* Business Unit Managers of all processes that include and follow KTM
* Future Validation Operators
* IT Project Manager
* Document Management Manager
* Scanning Team
* Backend System Manager
* Process/workflow Manager

This meeting is often the first time that all of these people have sat down together to discuss this project as a team. Expect the meeting to be loud and emotional. Often people think you are just delivering a piece of software, but in this meeting they realize you are changing how they work every day… and it is important that they reach consensus and own the solution.

Your KTM project will not only reduce effort for the document team, but it should also reduce effort for many other people in the organization. Unfortunately, some KTM solutions are simple data-entry solutions and have failed to realize the potential to reduce effort across the organization.

### Which fields should KTM deliver?

Often your customer has told you “We need these 23 fields from these 50 documents”, however the “23 fields” are simply what the Database Administrator has exported from the database definition – they are not really the fields needed to drive all of the business processes.

Your goal should be to deliver the fewest possible fields to drive all following processes.

Ask the following questions for every field.

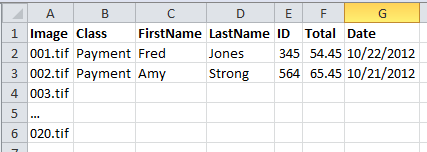
* Why do you need this field? What processes does it drive? Who needs to use it?
* What would happen if you didn’t have it?
* Can the field be found in a database or record elsewhere?

It is always better in KTM to know the field value and search for it on the document, rather than trying to find an unknown.

Ask your customer to explain, for every field, why it is needed.

* Agree on the value and goal of every field in every document class
* Ensure that the output of KTM is sufficient to drive all following processes.
* Ensure that the KTM project is minimizing effort by all people in following processes.

In the example below the Customer wants FirstName, LastNameand ID. You should argue that KTM delivers the field ID and not FirstNameor LastName. The field FirstName is not usable for driving a process. Is the customer’s name Fred or Frederik or Freddy or Frederika? Only ID=345 is useful. You should be arguing that the customer gives you the customer database so that you can look for the fields. You should agree with your customer that FirstNameand LastName will not be delivered by your solution.



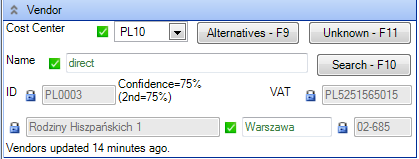
### Validation Fields

Some fields that you want to have in KTM may not be of interest to the customer.

Many invoice customers only want “Total” and “Tax” found on invoices, while KTM needs to find “Net0”, “Net1”, “Rate1”, “Amount1” and “Total” as well - to be able to check the mathematics and also to train the fields. Without these extra fields, KTM cannot possibly validate the required fields.

The Customer only needs the VendorID from KTM and no other Vendor fields, but they are important to the KTM project for the database locator, and also for the field viewers so that the user can easily validate the vendor.

Validation Fields are useful for

* Locators that need extra fields to locate, validate and learn (Database Locator and Amount Group Locator)
* Providing extra fields, fieldviewers and labels in the Validation Module to help the user make choices fast.  
  

### Agreeing on your Data

In this meeting it is important that you have agreed with your customer completely on these items:

* Representative Data Set for all fields, classes, exceptions, and processes that this solution is supposed to help.
* Which fields are exported from KTM.
* Validation fields that are used internally by KTM.
* All Databases are available for building the project.

If this meeting is successful, your customer will have much more confidence in your ability to deliver, and your implementation process will be smooth.

This meeting should be heated and passionate. You want to remove as many fields as possible from the project. Every field needs arguing on its own merit. Someone will insist FirstName is important – you need to convince them that it is not, and that you need the customer database connection. Here is also where you define and agree with the database admins exactly which databases, and columns are available.

## Writing your Statement of Work

At this point you can offer:

* We will build a KTM project that will improve the productivity of operators in processing the documents that are in the representative test set by helping them to produce precisely the output that is in the Excel sheet.
* This project will be objectively measured on documents/person/hour.
* Metrics like field accuracy and character accuracy will not be used, because they do not contribute directly to improved productivity and can lead to wasted development effort.
* This project will be built, tested, and measured on the representative Test Data set that has been agreed upon.
* In the production system, you can have the reasonable expectation that every document similar to the Test Data is machine readable and trainable, and should not require human interaction.
* Any documents in production or user acceptance testing that are not like the Test Data, are out of scope and not part of this statement of work.
* During the development process, we will provide, at regular intervals, benchmarks on test sets to provide objective and meaningful data about the performance of the project. These benchmarks will be used to prioritize development and configuration tasks, improve communication and understanding, prevent regression of quality, but not used to set arbitrary goals, which may or may not be attainable.

When the customer has signed this, you are now ready to sit down in front of Project builder and start!

# Metrics

A customer naturally wants to test and measure Kofax solutions. Unfortunately, they often apply criteria that are not appropriate and can lead to false security and misleading results. It can even force Kofax to cripple the solution to win a scoring test, rather than demonstrate a solution with the best return on investment.

Good metrics give excellent insight into the human effort required. Unfortunately some metrics do not do that. **Accuracy** Metrics are popular with customers (especially technical people), but they are generally counterproductive. **Performance** Metrics are easier to measure and calculate ROIs with. Performance Metrics focus on the problems the people have, while Accuracy Metrics focus on a perceived technical solution.

## Accuracy Metrics

### Character Level Confidence

Here a customer asks how many OCR characters are above a particular confidence level. This measure is not helpful because various OCR engines score themselves differently. You also receive no information about whether the characters were right or not, or whether confident characters were right. Character level confidence also has a low-level focus on the input data as it appears on the document. This is not always the data you want to export and may or not even be process-relevant.

### Character Level Accuracy

This metric is usually found in data entry solutions where keystrokes per hour are measured. This is also a poor metric because there are different costs to different types of fields. A date field with an OCR error is usually detected when the date is invalid, or is far in the future or past. An amount field with OCR errors is detected because A+B<>C, and in some of these cases numerical values can be automatically repaired. An error in an address, VAT code, or city name is of no consequence for vendor detection. But an OCR error in an invoice number is very serious indeed.

A character level accuracy of 94% gives us no information about the human effort required to correct the other 6% of errors. KTM can correct or ignore many character-level errors by the use of dictionaries, databases, and formatting rules.

### Field Level Accuracy

This measure is an improvement over character level accuracy, because invoice number accuracy can be considered more significant than address accuracy. But here we still have the problem of how much effort is required to fix the errors. We have no insight into how many errors are in a field – often it is one character, sometimes two, and perhaps (rarely) three. We have even less insight into the effort required to correct such fields. As we will see later, correct or incorrect doesn’t tell us the full story or what the dangers are – see just below about false positives.

### Precision/Recall

This is a metric from pattern recognition and information retrieval, <http://en.wikipedia.org/wiki/Precision_and_recall> . It suffers the same failings as field level accuracy, because it does not tell us enough about the problems and how much effort is required to fix them. It also does not put a high significance on some types of errors – it simply ignores them.

If precision<100 then documents entered the backend system with incorrect data – a catastrophe.

If recall<100%, someone needs to validate more documents by pressing Enter. This is not a catastrophe, just extra work. This metric gives us no insight into how much data entry (true negatives) the user has to do.

The biggest weakness of precision/recall for KTM is that for most projects each and everydocument is very important for the business, and a precision less than 100% is a business catastrophe. For statistical projects in which only aggregate data, not each document, is important – and perhaps document validation is not even required – this metric can be useful.

### True/False Positives/Negatives

This is the best accuracy metric for KTM because it gives deep insight into the user experience, is easy to understand and visualize and is relevant for any method of field extraction.

These metrics can be used to make objective decisions to improve the quality and performance of a KTM project:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Type** | **KTM Status** | **Consequence for User** | **Possible Causes** |
| True Positive | Correct and valid | Field is correct and the user can ignore it. Every customer wants 100% true positives. | * good KTM Project |
| False Negative | Correct and invalid | False alarm. The user must press Enter. If this happens too often, the user is frustrated and can cause false positives due to lack of attention. | * low confidence threshold * doc similar to others but needs training * locators producing too many alternatives * incorrect confidence from script locator |
| True Negative | Incorrect and invalid | User has to correct the field or type data into an empty field. Data entry fields are true negatives. | * Unknown document * OCR Errors * Poor quality vendor database * Data is not on the document |
| False Positive | Incorrect and valid | Incorrect data leaves KTM Disastrous, user loses trust in KTM, productivity loss, user rejects KTM. | * Badly trained documents * Field confidence thresholds too high * Poor locator strategy * Bad validation rules |

## Performance Metrics

These are the best metric for testing the performance of KTM validation operators.

### People per Project

This is the easiest metric to measure a project and the most powerful, because it focuses in the real problem – people’s effort and their costs. For example an invoice processing solution might have three people preparing documents, two people scanning, 18 people doing data entry into SAP, and one supervisor. That is 24 people.

If the company has the goal of reducing the FTE (full time employee) count on this project, the problem can be described as 24 FTE and the solution is possibly 12 or even 6.

### This number is easily convertible to ROI.Documents per Person per Hour

In the previous example, if the 24 people together process 20,000 documents per day, their performance is 833 documents/person/day or 111 documents/person/hour, given 7.5 work hours per day. This metric is useful if the goal is to double or triple the productivity of the users.

# The Priorities of a Transformation Project

With an understanding that the most important metrics for a KTM project are **people/project** and **documents/person/hour** and the **true/false positive/negative metric**, we can now list the priorities of any Transformation project.

1. Improve user productivity by improving the user experience.
2. Remove false positives to prevent false data leaving system
3. Reduce false negatives to reduce false alarmsfor users.
4. Reduce true negativesto reduce keying effort.

Here we see that problems with OCR and accuracy are really fourth priorities. Unfortunately, many KTM projects focus on this low priority, missing the value to be found at the first three priorities.

As you are improving your project, you should start at Step 1, and the go down a step when you have reached diminishing returns at the current step. You will unlikely get each step perfect and will have to compromise somewhere and move to the next level

# Benchmarking

Generally there are two kinds of benchmarks – relative and absolute benchmarks

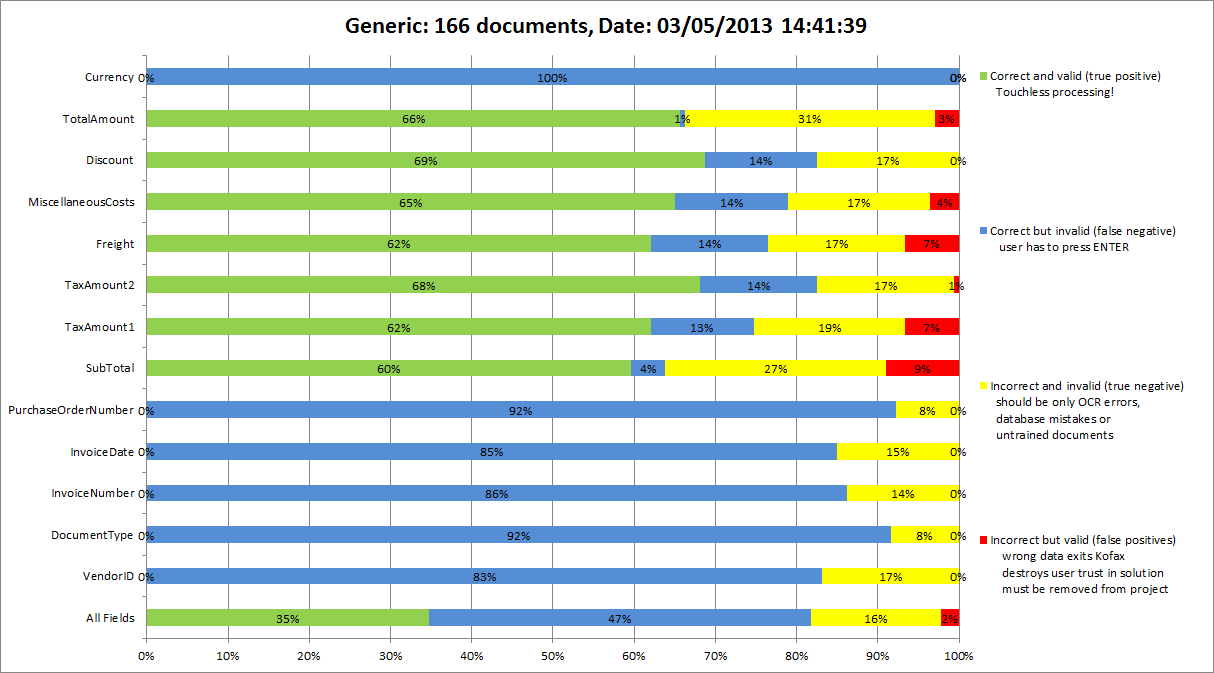
A **relative benchmark** is a score that you give to a system – and as you progress you keep testing yourself to make sure that it never gets worse than your benchmark – in this way you guarantee that your system is always improving and the benchmark is getting better.

Here, in the KTM context, we use an **absolute benchmark**. We know what the truth of the project should be by buildingCreating Golden XDocuments,. and are benchmark is an absolute and score of our system against the perfection of the golden files.

A single benchmark is an objective measure of how good a system is, and we can also reject a new version of a project if the benchmark gets worse.

In KTM we have Extraction Benchmarks, which measure field values, Classification Benchmarks, which measure document classification and also Separation Benchmarks.

Below is an example of a KTM Extraction Benchmark. It shows that the field TotalAmount can be skipped 66% of the time and that the user has to press Enter 1% of the time, they have to correct the TotalAmount 31% of the time, and that 3% of the time they skipped over invalid TotalAmounts.



## Value of Benchmarking

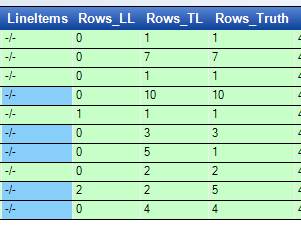
* A benchmark helps you focus effort on human productivity, not extraction.
* A benchmark shows you where you or your consultants need to put effort into your project next.
* You focus on all of your data, not on individual documents. This makes your project far more stable and robust.
* A benchmark communicates objectively to everyone the exact status of the project and the value of the work you did yesterday and this morning!
* A benchmark protects you from bugs creeping into your project, and you know the project is always getting better.
* A benchmark helps you find the best field confidence levels.
* You never have to look at “good” documents again. You focus effort always on problems.
* Tuning versus Tweaking
  + Tweaking is changing and testing one document at a time.Many people make a change to a KTM project, test document #1, test doc #2, make a change, test doc #2, test doc #3, make a change, test doc #4 and #3 and #2 and so on. This is a slow process and it means you have to look at your documents over and over again.
  + Tweaking makes a project unstable, creating tension within the settings.
  + Tuning is changing and testing a whole document set at one time. By testing all changes with a larger document set each time, you make decisions that are valid for sets of documents, not individual documents. This makes your projects more robust and probably faster!
* You become more disciplined and much faster in implementation. Benchmarking is guaranteed to cut weeks off your development time. With KTM 6 you can easily break a larger benchmark set of documents into smaller sets for faster running of extraction benchmarks.
* You can easily test if a new locator, script, checkbox, configuration, or slider is worth changing.
* Your project will have high performance on Day One of Go-Live.
* You have one simple graphical report, with objective numbers, that transparently and simply shows everybody involved the quality of the project.
* You can ensure that a project only improves in quality and never regresses (hence the term Regression Test).
* It helps you reach milestones and acceptance criteria, know when a new version is deliverable, and when your customer will pay you.

Don’t ever do a KTM project without benchmarking.

## Benchmarking Table Data

Project Builder doesn’t support Table Date Benchmarking out-of-the box. Below is an example of how you can partially achieve this.

In the picture below you can see the the LineItems cells are empty in the benchmark. The other 3 columns are hidden columns in the project that simply contain the number of table rows in the Line Item Matching Locator and the Table Locator. You can also have a hidden field showing the Table Sum



|  |
| --- |
| Private Sub Rows\_LL\_AfterExtract(ByVal pXDoc As CASCADELib.CscXDocument, ByVal pField As CASCADELib.CscXDocField)  pField.Text=CStr(pXDoc.Locators.ItemByName("LL").Alternatives(0).Table.Rows.Count)  End Sub |

## Training Data and Test Data

Training Data is used to build your project and Test Datais used to test your project. You will need at least 100 test samples in a Representative Document Collectionto be able to produce meaningful results. Your training data set should be even larger.

KTM 6 Project Builder has features to support training and test sets, including the ability to randomly divide a set of documents into test and training sets.

## How many Training Documents should I have?

The correct answer to this question is found at <http://en.wikipedia.org/wiki/Margin_of_error#Different_confidence_levels>

The simple rule for KTM projects is to use the following “95% confidence” formula.

**Example 1**.

A customer gives you 2 documents to build a project and tells you that they will test you on 3 unknown documents. You build a perfect KTM project on your two documents. You still have a 71% chance that your project will fail on the unknown documents – you are playing lottery and wasting your time. **Conclusion** – request 50 samples from your customer so your chance of failing drops to 14%

**Example 2**

You have 8 wrong invoice dates in your benchmark of 50 files (8/50=16%). There is a 95% chance that the invoice date on any other invoice being wrong is anywhere between 32% and 2% (16%±14%). **Conclusion** – your benchmark is too small.

**Example 3**

. A KTM project with 1,000 benchmark files, that have no false positives, still has a 95% chance of having up to 3.16% false positives in the productive environment. **Conclusion** – perfection is probably not possible.

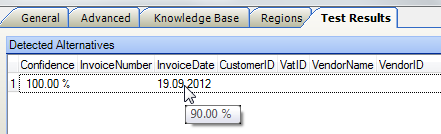
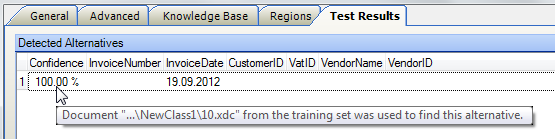
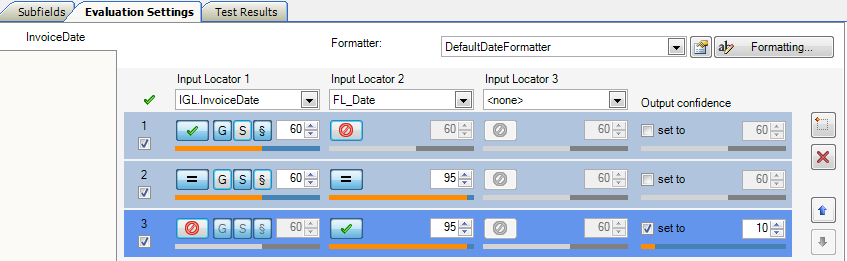
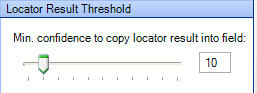
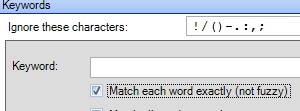
|  |  |
| --- | --- |
| Test Size | Margin of Error |
| 1 | ±100% |
| 2 | ±71% |
| 5 | ±45% |
| 10 | ±32% |
| 50 | ±14% |
| 100 | ±10% |
| 1,000 | ±3.2% |
| 10,000 | ±1.% |

This is why you should always be asking for 10,000 to 20,000 sample documents from your customer. This is completely reasonable when a good KTM project will produce this amount of documents PER DAY.

## How to fix mistakes found by a KTM Extraction Benchmark

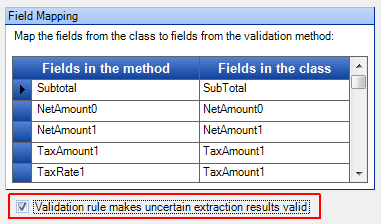
### #1 Remove False Positives

False positives hurt everyone. They need to be corrected first.

* Train documents properly.
  + Hover the mouse over subfields in trainable locators to see the trainable subfield confidence.  
    See [Trainer Locator Subfield Confidences](#trainerlocatorsubfieldconfidences) for the meanings of these confidences.  
    
  + Hover the mouse over the alternative confidence to find the training documents that caused the error in trainable fields.  
    
  + Format Locators mostly deliver values with 100% confidence. If these are wrong they will be false positives. Make sure you subjugate format locators to trainable locators in an Invoice Evaluator (requires Invoice Add-On Pack for KTM5.5, otherwise you can also script it. In KTM6 this locator was renamed to Advanced Evaluator and is license-free).)  
    
  + Here we see that the invoice date from specific training always wins. (We trust the user more than the locators!)
  + After that we can enforce that generic training must agree with the format locator
  + As a third option we accept the result of the format locator but give it a bad confidence (forcing it to be a blue or yellow). You can then experiment with the benchmark and these thresholds to optimize the field (no red and minimum blue). Start strict and then slowly relax the thresholds.
  + By default if the confidence is less than 10%, the field will not take the locator value.  
    
  + Format locators can achieve scores less than 100% if you allow fuzzy matching of keywords (and you should).)  
    
* Make sure that you have strict validation rules on every field.
* If you find a false positive created by a KTM locator or classifier, you should report it to Technical Support. Make a KTM project with one locator, one field, and one image that reproduces the problem. Report it as a KTM false positive to aid us in tracking.

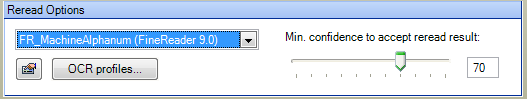
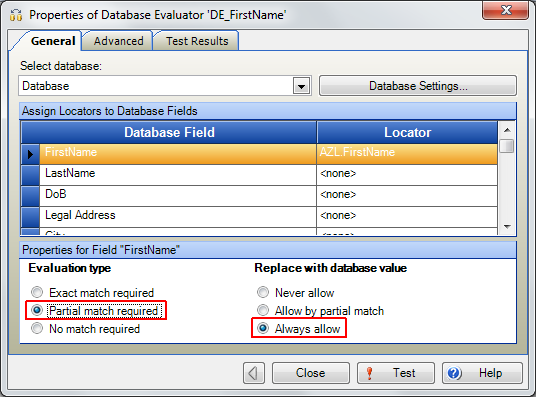
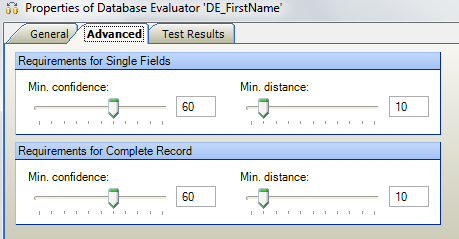
### #2 Reduce False Negatives

There will always be compromises between false positives and false negatives when fixing false negatives. Possible ways to improve false negatives:

* Modify locator thresholds and difference thresholds.
* Relax validation rules.
* Use multi-field validation rules so that some fields correct others.  
  

### #3 Minimize True Negatives

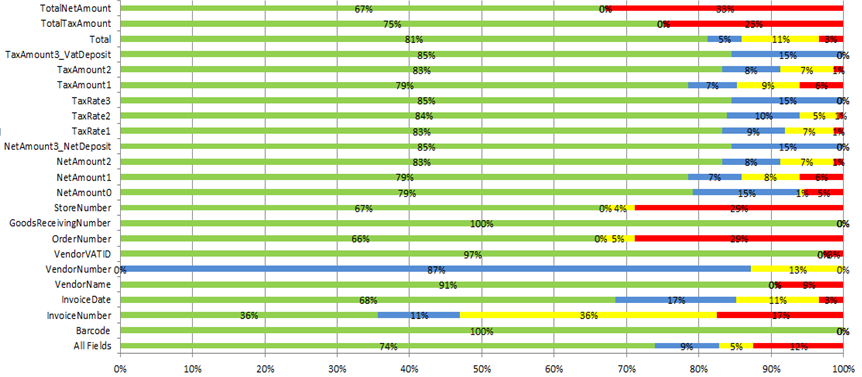
Remember that these errors are only priority #3 on your benchmark. You should be dealing with them only after the red false positives and blue false negatives have been resolved.

* Ensure excellent image quality.
  + Scan at 300 dpi. Optimize VRS Settings. Use Enhanced Clarity level 3.
  + Consider using Kofax Web Capture to build a web portal if customers are scanning & emailing invoices.
* Clean up Vendor databases. Use Vendor Locator to rescore Database Locator results.
* Use OCR Reread on the fields with stricter OCR Profiles  
  
* Spell-check fields using a Database Evaluator. This example spell-checks the FirstName field of a Zone Locator.  
  
* On the Advanced Tab you can adjust the sliders for when spellchecking occurs to optimize yellow vs. blue vs. red. Use the Extraction benchmark and lots of examples to find the optimum thresholds.
* Write a Database Formatter that does spell checking.
  + Create a script formatter.
  + Look up pField.text in a fuzzy database.
  + If the first alternative has a high confidence and the distance to the second is large, then auto-correct.
* Write a number formatter that does some OCR substitutions and then calls the DefaultAmountFormatter.

|  |
| --- |
| Private Sub RepairingAmountFormatter\_FormatField(ByVal FieldText As String, FormattedText As String, ErrDescription As String, ValidFormat As Boolean)  'Call the Default Amount Formatter  ValidFormat=DefaultAmountFormatter.FormatFieldText(FieldText,FormattedText, \_  ErrDescription)  If ValidFormat Then Exit Sub  'If it fails, try some cleanup  If Len(FieldText) = 0 Then FieldText="0" 'This works if "Require Decimal Point" is deselected  FieldText=Replace(FieldText,"l","1")  FieldText=Replace(FieldText,"I","1")  FieldText=Replace(FieldText,"O","0")  FieldText=Replace(FieldText,"o","0")  FieldText=Replace(FieldText,"S","5")  FieldText=Replace(FieldText,"B","8")  FieldText=Replace(FieldText,"G","6")  ValidFormat=DefaultAmountFormatter.FormatFieldText(FieldText,FormattedText, \_  ErrDescription)  End Sub |

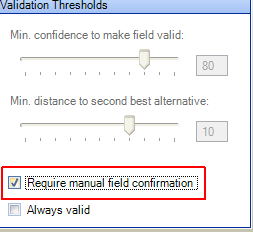
## Further Interpretation of a Benchmark

A benchmark can easily be used to analyze deep inside a KTM, just like an X-ray can see inside a person.



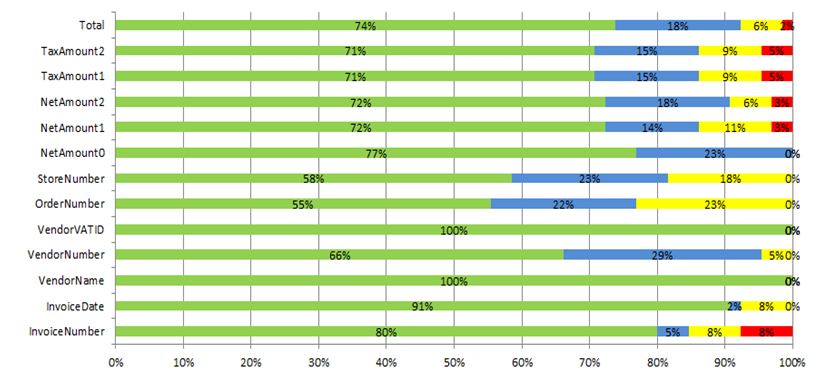
In this chart from a real project where the customer was very unhappy, we see that the fields TotalNetAmount and TotalTaxAmount are related to each other and that they are often wrong. But their benchmark is inconsistent with NetAmount0, NetAmount1, TaxAmount1 and TaxAmount2 from which they were *calculated****.*** This points to a logical error in a script locator – and should be easy to fix.

VendorNumber has far too many false negatives, which frustrated the users. But when we compare it to VendorVATID and VendorName, we see that there must be a configuration problem in the VendorLocator, or something else is causing this field to be always invalid. In this case it was:

.

Obviously the users had poor trust and demanded that each vendor be manually checked. However their mistrust was wrong, and they should have directed their mistrust toward the amounts fields.

Store Number and Order Number are clearly related and they came from a collection of script locators – there is probably an error in a script.

After a few days of work this was achieved. More work needs to be done to remove red false positives. At this stage we knew every single false positive in the benchmark set and why they all occurred.  
 

## Creating Golden XDocuments

You will need “golden files” to create a benchmark in KTM. A golden file is an XDoc that has been perfectly validated. The work your customer has done to provide you with representative data sets can enable you to quickly process golden files.

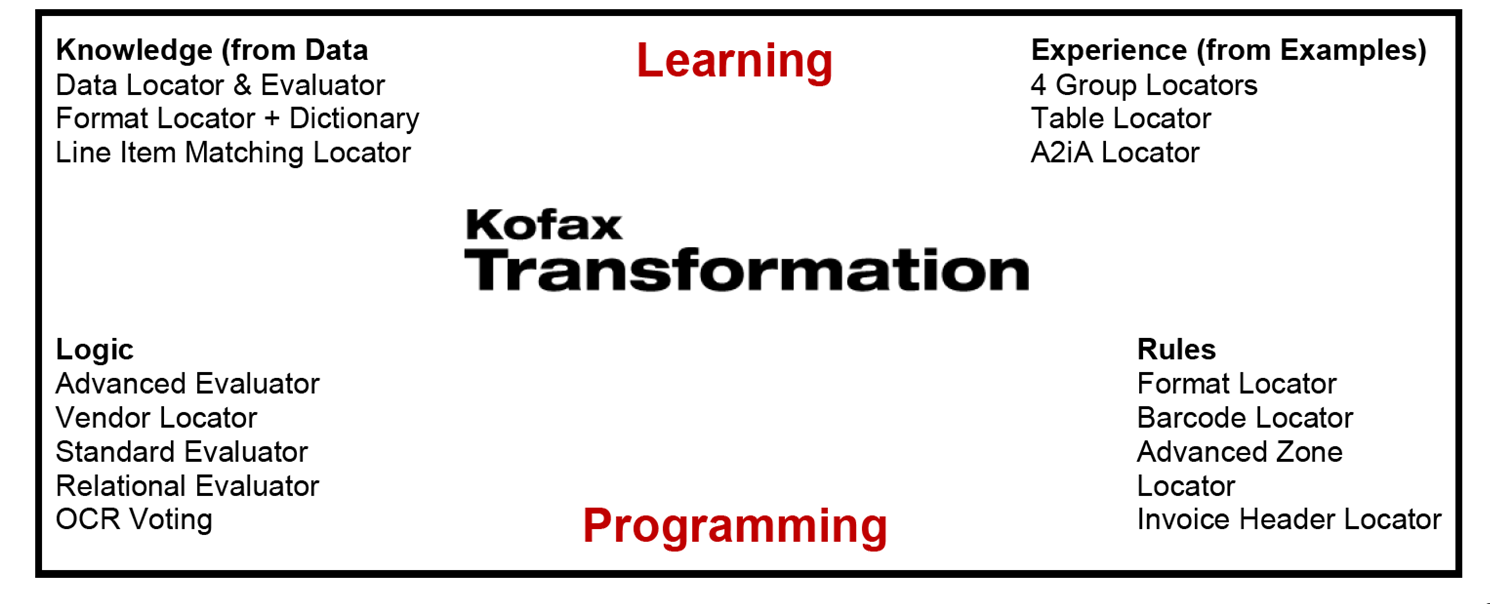
You can create Golden XDocuments in three ways

1. Build a KTM project without only classes and fields (no locators) and have someone do the data entry into Validation in Project Builder. Save the XDocs when complete
2. Use an existing KTM project in Kofax Capture, but turn off Kofax Export Connector. Copy validated XDocuments and TIFFS out of the KC Image folder.
3. See the Appendix [Create Golden Files from CSV Data](#creategoldenfiles) for a script on how to automatically create golden files from Excel data or extracts from ERP systems.

# Implementing a KTM Project

## Build a Project that learns

Your KTM should try to mimic a human as much as possible. Humans learn by experience and your project should also learn by experience. Use the following chart to decide which locators to use. Your project should use learning locators first, and use rules locators only as a backup. You then use Logic locators to subjugate rules locators to learning locators.



There are two ways a project can learn – from external data through databases and from training samples.

These locators can learn from external data and require some configuration:

* Database Locator & Database Evaluator
* Line Item Matching Locator
* Format locator + Dictionary. This is a magic combination! The format locator then returns all occurrences of each entry in the dictionary with fuzzy matching. Excellent for find phrases in documents.

These locators learn from training documents and require little configuration for learning:

* Invoice/Amount/Order/Trainable Group Locators
* Text Content Locator in KTM6
* Table Locator
* Line Item Matching Locator
* A2iA Locator learns from prebuilt A2iA check models.

These locators are rules-based and often require heavy configuration:

* Format Locator, Barcode Locator, Advanced Zone Locator
* Invoice Header Locator.(you should consider this locator obsolete)

The following locators are logic locators, because they do not locate by themselves. They evaluate the results of other locators.

* Advanced Evaluator.A very important locator that should be inside every KTM project controlling all trainable independent fields.
* Standard Evaluator. Made obsolete by the Advanced Evaluator, but can still be used for its simplicity. Do not use this for trainable fields because it does not distinguish specific from generic learning.
* Vendor Locator.This is actually a Database Locator Rescorer. Do not be misled by the name Vendor. It rescores the results of a database locator.
* Relational Evaluator and OCR Voting are rarely used.

## Steps in Building a KTM Project

Based on The Priorities of a Transformation Project we can now list the steps that you should follow when building a KTM project. The focus first is on user experience, and last of all on removing true negatives.

* Create the project
* Create the class structure
* Create the fields
* Load the documents and perform OCR.
  + Produce the golden files in Project Builder Validation  
    As you are doing data entry, go back to the Validation form and make these improvements:
    - Turn on **Autocomplete** wherever you can to avoid using the mouse.  
      
    - Change **Field** and **Field Viewer** sizes and font sizes to make values easy to read.
    - Build field **Formatters**
    - (*And, yes, build a few quick locators to make data entry go faster – but don’t spend time in the locators)*
    - **Database** lookups
    - **Validation** **Rules**
* Save your Golden Files.
* You have now proven that you have a good user experience, even before you have started on the Locators.
* Run the Ex**traction Benchmark** and fix the **red, blue** and **yellow** problems in that order.
* As the last step, we build the locators, . Locators are the last priority in building a KTM project. Locators are really solving the problem of true negatives, which we saw earlier were actually the lowest priority goals in a KTM project.

This is a very different approach from the way most people build their KTM projects (starting with locators). I strongly urge you to build the locators last.

Watch this 20 minute video <http://youtu.be/bIDmLVHmJNM> to see this process in Action!

## Clear Separation of Logic and Data

In much the same way that a webpage is made up of two separate components – content in HTML and style in CSS – your KTM project should also have a clear separation between Logic and Data.

Logic is the way your classification strategy, locators, formatters and validation rules work together.

Data includes the values that your logic uses. Data should be external to a project – KTM provides you with Script Variables, Script Resources, Databases and Dictionaries.

Script Variables are used to store parameters for your scripts.

Script Resources are strings for displaying to users that can be translated for localization.

KTM Fuzzy Databases can be used to store tables of values.

## Why you should avoid the Format Locator as much as possible

Format Locators are a tempting starting point for deterministic programmers as they are familiar and they make sense.

Before you use a format locator with regular expressions, consider whether one of these locators might be better:

* Trainable group locator for finding a phrase with an anchor
* FormatLocator+Dictionary for fuzzy matching
* Advanced Zone Locator, if the value appears always at the same location

However they have a number of weaknesses.

* You might be preferring a deterministic approach to learning. Your deterministic approach should be a backup to learning and subjugated to it.
* They require a long time to configure. Many format locators contain many regular expressions and evaluation settings. The effort that went into configuring these is often better spent training documents with a Trainable group locator.
* They can be slow because they perform regexes on every word of the document. Database locators and group locators are faster.
* They are not fuzzy (except when using dictionaries).
* You must expect them to fail. If they are feeding script locators, make sure your script locator handles the case when no result is found.

Regardless, they are still useful locators:

* For finding amounts for the table locator.
* For precisely matching Tax codes and other numbers in the Vendor Locator.
* Combined with dictionaries, they are powerful fuzzy phrase finders.

# User Experience

User Experience is the most important part of a KTM project. The story at the beginning shows that effort put here can be 10 times as effective as building locators.

Your first job in building your project is to create a **great user experience.** You also need your golden files as soon as possible, even before spending much time on locators.

Building your user experience first gives you many advantages

* You know your customer’s documents well and all their weird problems.
* You are focusing on the people first and not the technology and this sets the tone of the project form the beginning
* You can prove to yourself that the data is really findable from the documents.
* You hook into the project external databases as fast as possible
* You build a very good **data-entry** project very quickly. I am convinced many KTM projects bring great value to the customer simply by having great data entry, and not by have good locators.
* You get great golden files.
* You get all of your field formatters, validation rules and GUI design right at the beginning. The locators are behind the scene.

## Steps to building a great User Experience – Fingers , Eyes & Mind.

The User Interface is the most important part of your KTM project. You should spend much more time building this than in building locators.

Some of the tips below could improve your users’ productivity by more than 100%.

Here are some principles to following when designing your user interface.

### User Productivity

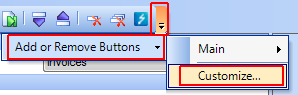
Every single decision you make should make the user more productive. Always be thinking “How can I make it easier, faster? How can I do it in less keystrokes?

You can help a user to be much faster by helping their **fingers, eyes** and **mind.**

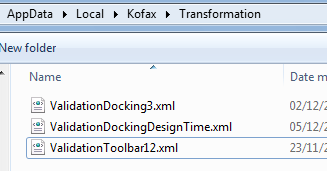
### Fingers

Make sure that the user can control everything with the keyboard and that the finger movements are comfortable.

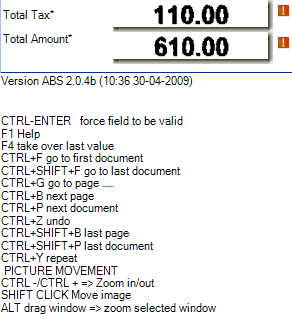
#### Use Hotkeys and show them to the User

There are hotkeys for almost every activity that you would do with the mouse!  


Custom Keybindingsare stored in **%appdata%\Local\Kofax\Transformation**

****

Put all of the hotkeys onto the Validation Form in a label, so that it is in front of the user’s eyes and easy for them to learn and use.



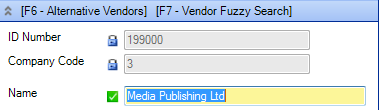
You cannot directly make multiple line labels with Project Builder. Use “\n” in the labels for line feeds, and the following script to convert to multiple lines. *(Make sure that ValidationForm scripts are attached to the class that the Validation Form belongs to, and not in the Project Script)*

|  |
| --- |
| Private Sub ValidationForm\_DocumentLoaded(ByRef pXDoc As CASCADELib.CscXDocument)  With ValidationForm.Labels.ItemByName("LabelHelp")  .Text=Replace(.Text,"\n",vbCrLf)  End With  End Sub |

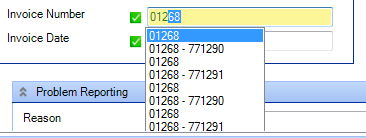
#### Use F6-F12 Function Keys for Buttons.

You also have CTRL and SHIFT. Be careful to space out function keys – F6, F8, F10 are better spaced than F6, F7 and F8.

A very good idea for buttons is to put them on an invisible Tab (ValidationForm.Tabs.ItemByName("Buttons").Visible=False) and only use labels to show the hotkeys. This forces the user to use the keyboard and also prevents buttons being left with the keyboard focus when dialogs are closed.



#### Use the **Autocomplete** Feature

This is a fantastic productivity feature in KTM, and so familiar to people from Google Search with its suggestion features  


Autocomplete was added to KTM to get field viewer coordinates for the online learning mechanism without forcing the user to use the mouse.

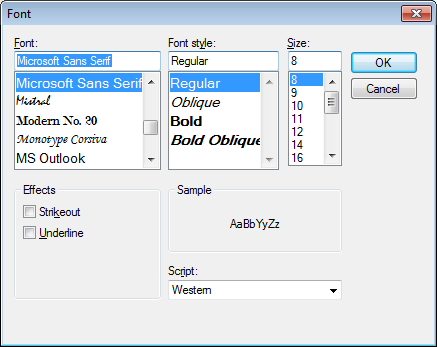
Below is a list of how to use it to ensure that field viewers have the correct coordinates for the fields – this is important if you want fast users who know how to train documents.

|  |
| --- |
| AUTOCOMPLETE  ENTER to validate  ESC accept snippet, but edit text  SHIFT + ESC accept text but not snippet  Press "SPACE" to select next word in autocomplete  Arrow up , shift-tab previous Field  Arrow down shift-down necxt field |

### Eyes

Guide the viewer’s eyes comfortably down through the fields. Don’t force them to have to glance back and forth between the document and the fields.

#### Fonts, Layout and Colors

Make your validation form layout pleasing to the eye. Your users will be staring at it all day every day. Use different fonts to make things easier to read.  


You can even dynamically change the color of a label in script. *(Make sure that ValidationForm scripts are attached to the class that the Validation Form belongs to, and not in the Project Script)*

|  |
| --- |
| ValidationForm.Labels(0).ResetForeColor  ValidationForm.Labels(0).SetForeColor(255,0,0) 'Red |

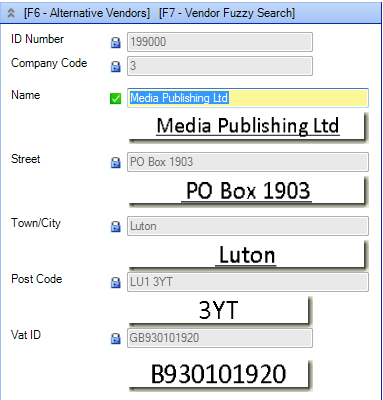
#### Field Viewers

Field Viewers serve two purposes

* Show the coordinates on the image for the online learning mechanism
* Help a user to quickly make a decision about a field

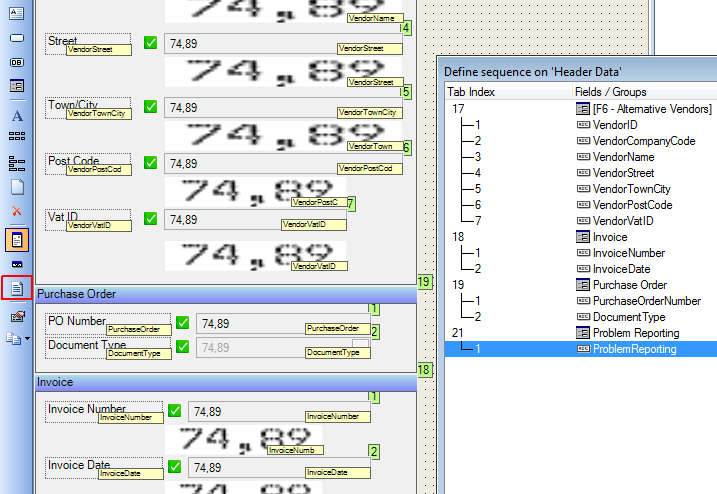
You can align field viewers underneath or above the fields to make it easier on the eye, when you have to check long numbers.  


In the example below you can see the use of Field Viewers and read-only fields to make it easier for the operator to decide “Do I press ENTER to confirm or DO I press F7 to Search?” when searching for a vendor. They can make that decision without having to glance to the document.



#### Field Sequence

Make sure you have configured the field sequence to step through the fields from top to bottom.



You must never allow the cursor to jump up to a previous field. Always place your fields on the document so that dependent fields are lower on the screen.

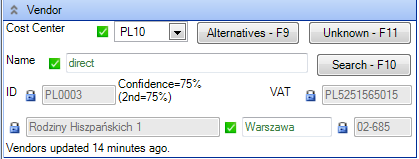
### Mind

The user is constantly confronted with a new decision every time the cursor is blinking in a field. Your job is to make it as easy as possible for the user to work out, “Is that a false alarm (false negative)?” and press ENTER or to decide, “That is wrong, I need to change it”.

Provide the user with as much useful information as possible to make it easy and fast. But provide the information passively, so that the user can always choose to ignore it or user it.

#### Passive Information in Labels

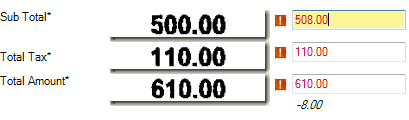
Passive information appears in labels that the user can choose to use or ignore. You can show **confidences, alternatives, discrepancies, database lookups**.

The example below shows the confidence of the Vendor Detection, as well as the confidence of the second alternative – this can help the user decide to either press F9, F10 or the ENTER key. You also see how many minutes old the Vendor Database is.  


Here is an example of numeric discrepancy. The label shows a **discrepancy** of -8.00 in the Total Amount.

In the **Thick Client** you can change labels with the **ValidationForm\_OnFieldChanged** Event**,** which is fired on every keystroke. In the **Thin** **Client**, you have to wait until **ValidationForm\_OnFieldChanged** when the user presses the ENTER key.

This helps the user find the problem that a “0” has been replaced by an “8” in the SubTotal

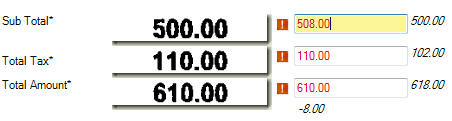


#### Suggestion Labels

We can often do even better than just providing passive information about problems. We can make suggestionsabout **numbers, alternatives** and even **Spellchecking.**

**SubTotal + Total Tax = Total Amount**.Here the user can see what all the numbers would be assuming the others were correct. (Make the suggestions go away completely when the fields are correct).

Here you have really helped the user by doing arithmetic for them, and drawing their eyes to where the problems are – and even before they have pressed the ENTER key, they know that the field is valid.



The user can now see quickly that “500.00” is needed instead of “508.00”, but still requires **FIVE** keystrokes to fix it **LEFT, LEFT, LEFT, BACKSPACE, 0.**

We can make this even easier for the user by having a hotkey for automatically copying the suggestion. If the user presses “?” then the correct value is copied. You can use the following code.

|  |
| --- |
| Private Sub ValidationForm\_AfterFieldChanged(ByRef pXDoc As CASCADELib.CscXDocument, ByRef pField As CASCADELib.CscXDocField)  If InStr(pField.Text,"?") Then  pField.Text=ValidationForm.Labels.ItemByName("LabelSuggestion" & pField.Name).Text  End If  ... |

#### Error Messages should suggest possibilities

Many field error messages contain phrases like **“This value is wrong”** or **“Total<>Subtotal + Tax”** or “**There are more than one alternatives”**, but these errors do not actually help the user find the answer – and often it is easy for you to know what the correct answer should be, or to suggest possibilities to the user.

Change your error messages to be more helpful like **“”Press ? for 500.00”** or **“Press ? for Vendor XYZ. Press @ for Vendor ABC”**, so that the user can fix problems with a single keystroke.

# Classification

## Do you really need to classify documents?

A customer might think they need to classify documents because they have been doing it manually. During the

Customer Walkthrough and Document Analysis phases, you had the opportunity to really understand what was happening with the documents. Here you need to really question the processes that are paper-based and find a good justification to transfer these processes to your Transformation Project.

Documents only need classifying in KTM so that you can apply locators to those documents. If your customer has 20 “classes” of documents but reads the same 5 fields from each of them, do you really need to classify them in KTM?

You need to find out how your customer names the documents, as well as how documents are grouped together.

Ask your customer “Why are documents being classified now?”

|  |  |
| --- | --- |
| Answer | Suggestions and further Questions |
| So that we can find them later. | Use Kofax Express to produce searchable PDFs without classifying and export to the customer’s Document Management System where you can do a text search to find them quickly, rather than manually hunting through a classification tree. Find out if a really simple solution might be better.  Your customer might simply be happy if all 50 pages of a customer’s file become one big PDF file and use the PDF viewer’s search function to look inside it.  Ask if anyone will ever look at these documents again. What is the cost of not finding the document again?If you are scanning millions of pages and only 0.1% of them will ever be looked at, why invest a huge effort? |
| Because we have a complex classification system we have been using for years | Maybe the system is out -of -date, inconsistent and more than needed. Find out if a simpler classification strategy might drive all the required process |
| Because different documents need to go to different departments | If there are only four departments, then classifying into 40 classes might be excessive. |
| Because we need to extract different fields from different documents. | Maybe you only need to extract a few fields from a folder of many document types. Instead of classifying all the documents, just classify the particular pages where info needs to be extracted and leave all other pages as “default” with no locators and fields. |
| Because we are doing business intelligence on the documents | Each document is unimportant by itself, only the aggregate results matter. This means you have a higher tolerance for false positives. You may not need document review and validation module at all. You also only need to classify to the level of the statistics you are generating. |

## Classification Trees and Field Inheritance

You need to define your fields in KTM at the highest level possible in the class tree to benefit from the inheritance mechanisms for formatters, validation rules and locators.

Below is the result of field analysis on some Portuguese Documents.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **File Name** | **Class Name** | **Field Names** | | | | |
| **Capa** | **Nro\_DOC** | **NOTA** | **CPFC** | **EMIS** |
| **1.tif** | CapaLote | 123987-2012 |  |  |  |  |
| **2.tif** | Duplicata | 123987-2012 | 852147-A | 60.000,00 | 07.248.659/0001-03 | 15/02/2013 |
| **3.tif** | Duplicata | 123987-2012 | 1489/1 | 15.963,57 | 17.155.342/0003-45 | 01/12/2011 |
| **7.tif** | Nota Fiscal | 123987-2012 | 65357 | 7.981,39 | 80.089.964/0001-97 |  |
| **8.tif** | Nota Fiscal | 123987-2012 | 194.580 | 48.741,92 | 76.777.556/0001-50 |  |

|  |  |
| --- | --- |
|  | This corresponds to the Class Tree in KTM on the left, where the field “Capa” is defined in the class “CapaLote”, and “Nro\_Doc” is in the class Duplicata. |

## “Structural” Classes

These are classes that exist in projects, but they don’t have any documents associated with them.

|  |  |
| --- | --- |
|  | Sometimes you need to insert “structural” classes into a class hierarchy to hold fields and validation rules.  In this example you can see three structural classes “Document”, “Application Forms” and “FormA”. They have the following classification features disabled  “Document” may not even have any fields and no documents will be classified to this class. The same is true for “Application Form” and “FormA”. “FormA” has a few variants that require different locators but they inherit fields from FormA.  Fields that are in common between FormA and FormB are defined in “Application Forms” |

Structural classes are also used in the classification mechanism to resolve ambiguities. If a document had an 85% score for “FormA 1.0” and 83% score for “FormA 1.1” then the document would be classified as “FormA” if you allowed “Valid classification result”.

Your project should always have a top-level structural class called “Document” to keep reminding you of the inheritance mechanism and also to keep yourself future safe – as your project grows you will need to add more classes.

**Separation and Classification always go hand in hand - your strategy for one will affect the other**

## Building Training Sets

Your customers have deep knowledge of their own documents and know how to do them. They should be responsible for classifying documents, and naming them, rather than you or a project manager. Use as much of the customer’s knowledge as possible in your project. That means teaching your KTM project about classification, preferably with hundreds and thousands of documents that your customer has already classified.

## Classification Tools

Here is a list of classification “tools” listed in the order that you should consider using them.

### Extracting automatically from a Document Management System.

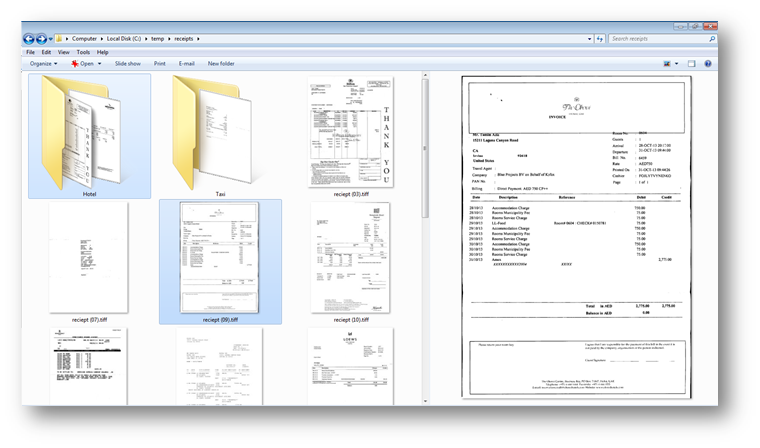
This is the best method of building a set of classification documents because the work has already been done by your customer. In any classification project always ask about projects that have already been classified, and use them as a classification basis.

### Windows Explorer

**Windows Explorer** has great value as a classification tool. It is installed on every machine and your customer knows how to use it intuitively. It is very visual, and more than one person can classify documents at the same time. You can classify many hundreds of documents per hour and easily check and correct for classification mistakes. If someone has a large monitor then they could even classify 10 or 20 documents at once by dragging them into a folder. One Customer in the Netherlands classified 3500 documents over two days using Windows Explorer.

It is the most useful tool for your customer during their document analysis phase to **understand, organize and name** their own documents.

**Settings**

* View\Extra Large Icons
* Organize\Layout\Navigation Pane to hide it – you could also use the navigation pane if you have many classes with a tree structure.
* Organize\Layout\Preview Pane
* Keyboard Shortcut to create a new folder CTRL-SHIFT-N

### Build a KTM project simply for Classification

Windows Explorer is likely to be simpler and faster for the user than trying to build a custom solution. The user cannot add new class names, and KTM validation requires the user to select the Class Name from a drop-down list with the mouse and then press ENTER – it is not built for speed.

The best solution would be to have a single field containing the classification name and apply a script format locator which uses a fuzzy database locator to choose the correct class name. The documents can then be reclassified by using a special script later.

### Document Clustering

This was a standalone tool in KTM5.5, and has now been integrated into Project Builder 6.0.

This is not a good tool for helping a customer to understand their documents, because it does not help with naming and organizing – the user is confronted with a document and mustname it. There is no option to defer a document or come back to it later.

Document Clustering is good if the customer has a very clear understandingof the documents and knows exactly how to name them.

Another disadvantage of this tool is that it requires the user to understand Project Builder in KTM 6.0.

### Trainable Document Separation (TDS)

Trainable Document Separation requires an immense amount of work and skill.

It uses <http://en.wikipedia.org/wiki/Hidden_Markov_model> to build training models to classify pages and separation documents. It requires thousands oftraining samples to be able to produce accurate results (read about <http://en.wikipedia.org/wiki/Sparsity> in Hidden Markov models for the reasons why)

You should only use Project Planner in KTM5.5 or Trainable Document Separation in KTM6 if the following criteria are fulfilled

* You genuinely need to classify and separate every single document accurately.
* There are FAR too many documents being processed to apply barcode stickers or separator sheets, or to use Document Review for Manual Separation.
* Your customer’s document experts can pick up any random page and easily tell you “This is a first/middle/last” page. If they cannot tell this, then TDS will possibly also be confused.

### Project Builder

Project Builder itself can be used for classifying by dragging documents into classes, retraining the project, classifying the remaining documents and sorting them by classification confidence. This way you can rather quickly classify (this process is quite similar to how the clustering tool would work, but you can stop and start as you like and choose which documents to work with)

## Classification Methods

Kofax Transformation has seven methods for classifying documents at runtime.

### Layout Classification

This method is fast and can classify hundreds of pages per second without requiring OCR. Layout classification is mainly used for fixed forms, specific training for invoice vendors. It is also used in Kofax VRS Auto-profiles and Kofax Express Online Form Learning.

### Text Classification

This method requires the OCR text of the document. It does not use any geometry. If matches letter groups from three characters up to about 25 characters per phrase. It is very good at finding unique letter patterns that identify a page. It is good at classifying unstructured documents, email bodies, and even fixed forms.

It is probably not good at classifying invoices because they share a lot of common vocabulary, and table line item descriptions are mostly unique on each document.

In KTM 5.5 you can choose whether a training document uses layout or text classification. In KTM 6 this choice is made automatically.

### Instructional Classification

This method is simple to configure. You need only to give it phrases and a score from -100% to 100%, but it is a deterministic methodology and contains a number of risks. It is easy for it to make false positives and may require a lot of work to be trustworthy. It doesn’t consider the location of the phrases on the page. See Script Classification for searching for identifying phrases inside or outside particular regions of a page.

Avoid instructional classification unless no have no other choice, or unless there are very few document classes.

### Script Classification

Use script classification for the following cases:

* Classification by barcode value, database lookup, scan operator name, phrase at a particular location on the page, and so on.

See Appendix Visual Basic 6 tips

Use With … end With to make your code readable.

Use For…Next instead of copy/pasting code.

Use the following to loop through “Total”, “SubTotal”, etc…

For each fieldname in split("Total SubTotal TaxRate1 TaxAmount1")

Do error checking at beginning of functions.

If condition then exit sub.

Use the setting '#Language "WWB-COM" to get keywords return, andalso, orelse, isnot.

All four keywords will reduce nesting and increase readability.

Have tight logic loops and functionalize what is in them for clarity and easy debugging.

The following example reads every word on a line, fuzzy matches it to a database and then builds a score. It is quite complex, but the complexity has been pushed off into another reusable function, and this loop is now very simple.

|  |
| --- |
| For w = 0 To textline.Words.Count-1  word=LCase(Trim(textline.Words(w).Text))  match=**DataBase\_SearchString**(databaseName,"headerword",word,conf)  score=score+conf\*Len(word)  Next |

## Fuzzy Database Queries from Script

This is a useful and flexible script for retrieving one or many, fuzzy or precise results from a local or remote fuzzy database.

Combining an SQL Database with Kofax Search & Match Server and this script you can perform webqueries much faster and even perform fuzzy “webqueries.””.

|  |
| --- |
| Public Function Database\_Search(dbname As String, column As String, searchString As String, numberHits As Integer, minimimConfidence As Double, Optional allColumns As Boolean=False) As CscXDocField  'Searches inside a fuzzy database for the searchstring and returns the results in the alternatives of a new CSCField Object.  'if column="" then all columns are returned as subfields, otherwise returns only the chosen column in the alternatives.  'Set minimimConfidence=1.0 for exact match search.  Dim DB As CscDatabase, Fields() As String,FieldIDs() As Long  Dim col As Integer,c As Integer,i As Integer  Dim hits As CscDatabaseResItems, alt As CscXDocFieldAlternative  Dim results As New CscXDocField 'You are allowed to create a standalone field  Dim value As String, substitute As String  Set DB=Project.Databases.ItemByName(dbname)  ' Replace all delimiters by blank  For i = 1 To Len(DB.AdditionalDelimiterChars)  searchString = Replace(searchString, Mid(DB.AdditionalDelimiterChars, i, 1), " ")  Next  ' Replace all ignore characters by blank  For i = 1 To Len(DB.RemoveChars)  searchString = Replace(searchString, Mid(DB.RemoveChars, i , 1), " ")  Next  ' Substitution pairs define which texts to be replaced by what.  For i = 0 To DB.SubstitutionPairCount - 1  DB.GetSubstitutionPair(i, value, substitute)  searchString = Replace(searchString, value, substitute)  Next  Fields = Split(searchString, " ")  ReDim FieldIDs(UBound(Fields))  'Find the column we are looking for  col=-1  For i =0 To DB.FieldCount-1  If DB.FieldName(i)=column Then col=i  Next  If col=-1 And column<>"" Then Err.Raise 34589,,"Column '" & column & "' does not exist in database '" & dbname & "'."  If col<>-1 Then 'Force query in this column  For c=0 To UBound(FieldIDs)  FieldIDs(c)=col  Next  End If  Set hits = DB.Search(Fields, FieldIDs, CscEvalMatchQuery, numberHits)  For i = 0 To hits.Count-1  If hits(i).Score>= minimimConfidence Then  Set alt= results.Alternatives.Create()  alt.Confidence=hits(i).Score  If allColumns Then 'the column is "", so we return all fields  For c = 0 To DB.FieldCount-1  alt.SubFields.Create(DB.FieldName(c))  alt.SubFields(c).Index=c  alt.SubFields(c).Text=DB.GetRecordData(hits(i).RecID)(c)  alt.SubFields(c).Confidence=hits(i).Score  Next  alt.Text=""  Else  alt.Text=DB.GetRecordData(hits(i).RecID)(col)  End If  End If  Next  Return results  End Function |

How to Classify by Barcode in Script for an example.

and How to Classify by Phrase in a Region on a Document for examples.

### Classification Locator

You can use the Classification Locator to simplify complex classification, by using other KTM projects to classify certain groups of documents, and then use script classification to apply the classification results in the main project

### Manual Classification (Document Review & Validation)

If documents require a lot of human insight to classify, then just use Document Review to allow the users to classify the documents.

Manual Classification in Validation is not good for user productivity and should only be used as a last resort.

### Trainable Document Separation

Trainable Document Separation requires a great amount of work and skill. You should use only **Project Planner** in KTM5.5 or **Trainable Document Separation** in KTM6 if you genuinely need to classify and separate every document accurately. See Trainable Document Separation (TDS)

## Classification Tips

### How many documents should I train for Classification?

You should train as few documents as possible for each class to keep your training sets small, fast and controllable.

Use the following technique to find the best training documents for a class. This technique guarantees to find the fewest number of documents that account for the greatest variation amongst your samples

1. Pick a document which is clean, has excellent OCR and looks like lots of others in the class. Add it to the training set.
2. Train the project.
3. Classify all the documents from that class and sort the documents by classification score.
4. Pick the document with the **worst** score and add it to the training set.
5. Repeat steps 2-4 until all of the documents have “good” (80% or more) classification confidence.

### Customer Classification is not always KTM Classification

Your customer might have a way of classifying documents that are logical to them, based on what the documents mean. In Kofax Transformation, however, you need to classify documents based on how they look and what phrases are used.These might be very different. Use class redirection to move a classified document to a completely different place in the classification tree.



### Divide and Conquer

Some pages might appear in multiple documents, perhaps as a last page of one document and a middle page of another document. Perhaps some pages are just attachments to a previous document. Classify each as **separate** document, and after classification has been performed, merge the document parts back together again in the Batch\_Close event in KTM Server.

### Find and Filter

You might find that some of your documents misclassify and that any effort to classify them correctly leads to an instable classification system. Let the documents be misclassified, and then in the Document\_AfterClassify use the CSCDocument.Reclassify(,**)** method to reclassify the documents.

# Document Separation

Kofax Transformation classifies individual pages and then separates them into documents.

There are seven different ways of separating documents – each method is valid in different scenarios.

## Use Barcode Stickers

This is simply the best method – fastest, cheapest, and safest in most scanning scenarios,

Every document has its own unique ID from the barcode value. Invoices and most other documents have some white space somewhere on the page where you can stick a barcode. For the very rare cases where that is not the case, stick the barcode on the back of the document and scan the document upside down, or stick the barcode on a separate piece of white paper as a cover sheet.

The paper document is then easy to find in the archive-box. Easy because the barcodes are unique and increasing.

During document preparation, each document needs unfolding, unstapling, removal of advertising, and perhaps page sorting. This paper handling stage requires a minimum of two to five seconds and the effort of sticking on a barcode takes less than one second. The user now has the guarantee of perfect separation.

If a customer designs barcodes according to Kofax specifications (three pixels per thin line and adequate quiet zones around the barcode), misreads should be in the order of one in tens of millions of documents.

**Handling Attachments**

Many customers use two barcode stickers,and the second is to tell Kofax Transformation to ignore the page because it is just an attachment, and there is no need to pay for the page-count in Transformation.

Most Kofax BPO customers and many invoice processing customers use barcode printers like those found at [www.zebra.com](http://www.zebra.com).

I have convinced every customer of mine to use barcodes, even those who were initially resistant to using them.

If you stick a barcode on the invoice and it is torn or folded, just stick another one over it or elsewhere.

## Scan documents individually

This is the best method for untrained users who only occasionally scan documents, and only minimal effort is needed to manually separate two or three documents.

This method is recommended for Kofax Web Capture and Kofax Mobile Capture solutions.

## Use Separator Sheets

Separator sheets are best in very large scanning solutions where there are ten to fifteen document classes and the customer wants to manually classify them.

Separator sheets also have a number of disadvantages because they double scanning time. About 50% of what you scan is not needed. Scanners also wear out twice as fast and need more maintenance.

Reusing separator sheets also requires a lot of work to separate them again, and the documents in the physical archive are not in the same state that they were in when they were scanned.

If you need to find the original paper again, you will search longer than if it had been barcoded.

## Combine all three methods

Physically separate all single-page documents and scan them with a “single page” separation profile.

Scan all the two-page documents separately with a two- page separation profile. In Total Agility 7 you also have single-sheet and double-sheet separation profiles.

Scan the remaining invoices with separator sheets, manually or with barcodes.

## Rubber Stamp with Datamatrix barcodes.

Make sure that the stamp is designed by an expert. A good datamatrix barcode has less than 75% redundancy. Datamatrix is the most robust 2D barcode type, superior to QR code in density and readability.

Stamp this barcode anywhere on the document. If the stamp doesn’t look good, just stamp again.

The only known weakness of this method is that some customers use these stamps too long before replacing them, which can lead to failed barcode reads.

## Trainable Document Separation (TDS)

See Trainable Document Separation (TDS) for more details. TDS is valuable for massive scanning projects in banks and insurance companies where other methods of separation are too slow, there are many document types, and the customer has some tolerance for mis-separations.

## Automatic Document Separation for Invoices

There is no good and robust strategy for automatically separating invoices. Recommend to your customer to use barcode stickers.

You cannot use TDS for invoices because invoices don’t have first, middle, and last pages that look different from each other. Invoices share a lot of the same vocabulary except for line item descriptions that vary.

An automatic separation strategy would have to deal with a vendor sending ten invoices together and the case where a delivery note is attached to the invoice and they look the same.

Any strategy for separation or combination of strategies (such as change of vendor, date, amount, PO, layout) has scenarios where this does not mean separation.

The risk to a customer of automatic invoice separation is very high because an invoice will not be paid! The effort to prevent this problem is more effort than the one second required to barcode the invoice.

This request typically arises where workers are scanning a small group of invoices at the end of the month with simple scanners that produce PDF files which are then emailed. Here a Kofax Web Capture solution would solve the document separation problem, the image quality solution, by putting VRS into the user’s browser and integrating the invoice-senders directly into the process.

Some customers solve this by charging the vendor a processing fee for doing manual separation in Document Review Module.

# Locators

Locators are at the heart of Kofax Transformation.

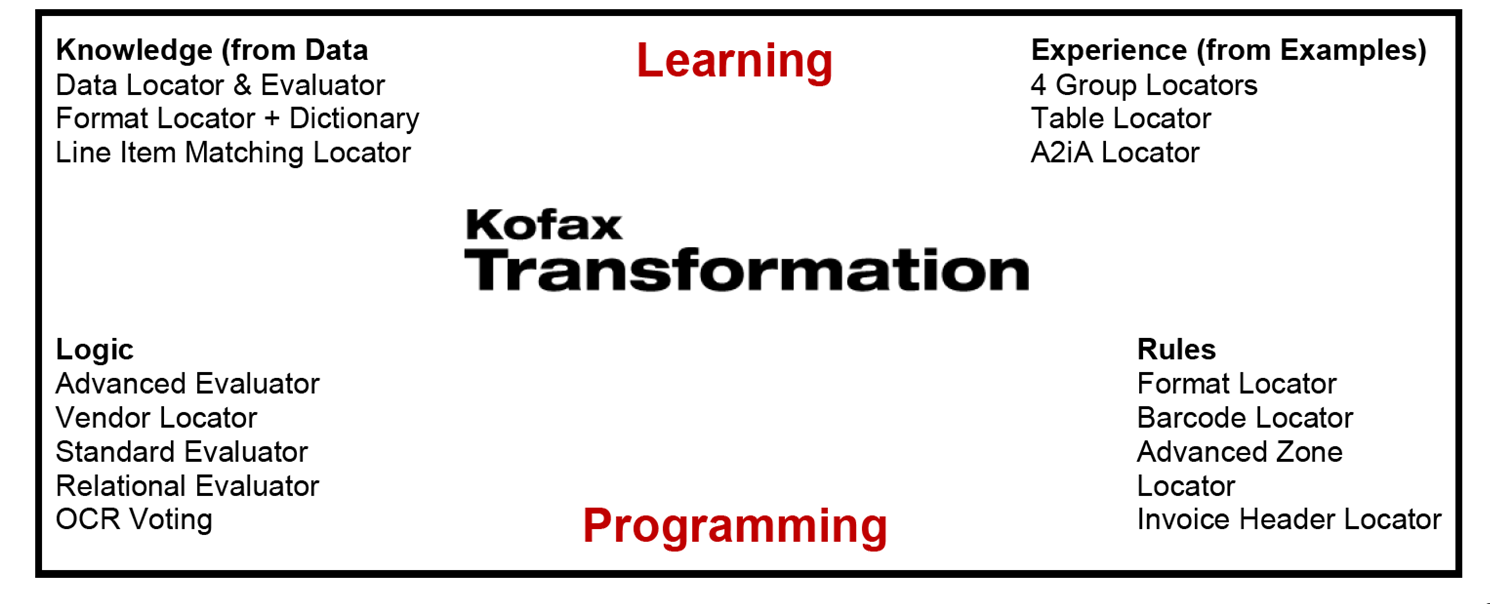
Locators use the document or other locators to find values and return coordinates and confidence of each alternative found.

Most locators can be restricted to look at or avoid regions on first, middle, or last pages of a document.

A single field locator returns alternatives with single values.

A multifield locator returns alternatives with multiple subfields.

A table locator returns 1 Alternative with a single table.

Locators can grouped into two broad categories, learning locators and programmable locators. You should prefer teaching locators to programmable locators because they require less effort to configure and can improve with time.  


There are two kinds ofLearning Locators:

* knowledge locatorsuse information from the outside world via databases and dictionaries to locate information on documents.  
  (*Here is where Kapow can be of immense value to KTM, because it can bring the internet into a project)*
* experience locators use examples of previous documents from the customer to interpret new documents.

There are two kinds of Programmable Locators:

* rules locators do not use outside knowledge - they are programmed or configured.
* logic locatorscombine the results of other locators together to produce alternatives

Build your project with knowledge and experience locators mainly, using rules locators as fallbacks, and then combine results together with logic locators.

**Teachable Locators** are easier and faster to configure than **programmable locators.**

|  |  |  |  |
| --- | --- | --- | --- |
| Knowledge locators |  |  | Use Cases |
| Database Locator  (multifield) | DL | Matches entire database with a document fuzzily  Finds only one occurrence of each DB entry | Finding vendors, customers. |
| Database Evaluator  (multifield) | DE | Matches an entire database with output from a locator fuzzily or strict  Finds only one result. | Used to spell check handwritten fields from AZL. Look up a customerID fuzzily/strict and return from the database the entire customer record.  Can function as a “fuzzy webservice call” script free. |
| Format Locator + Fuzzy Dictionary  (single field) | FL | Fuzzily find of a list of phrases on a document.  Finds all occurrences of each phrase | Find keyphrases to identify a document at a particular location on the page.  Find specific labels for fields |
| Line Item Matching Locator  (multifield) | LL | Fuzzily matches purchase orders from a given Vendor in an SQL database with an invoice | Matching each purchased item with the invoice. |

|  |  |  |  |
| --- | --- | --- | --- |
| Experience locators |  |  | Use Cases |
| Invoice Group Locator  (multifield) | IGL | Find a number and nearby date on a document. Return document ID if trained. | Finding invoice number, invoice date & retrieving VendorID from user training.  Finding CustomerID and ApplicationDate on insurance documents. |
| Order Group Locator  (multifield) | OGL | Find a number and nearby date on a document | Finding order number and order date on an invoice.  Finding delivery date and delivery number on a waybill. |
| Amount Group Locator  (multifield) | AGL | Find subtotals, taxes, total & currency on a document | Finding amount footer information on an invoice. |
| Trainable Group Locator | TGL | Finds a set of subfields that appear together on a document.  If you have unrelated fields, use more than one TG  Works well on fields with a clear anchor phrase | Finding customer Ids, special dates. |
| Text Content Locator  (multifield) | TCL | Find tokens in punctuated (?!,.:;) text on a document. Simple, trainable,fast  Ignores word wrapping  No rules  New in KTM6. | Find all occurrences of names, dates, addresses, ids, and so on, in a very large contract. |
| Table Locator | TL | Find tabular data on a document. Has three modes that all require training (manual, automatic and training) | Invoice line items.  Tables with rich numeric data  Only finds one table on a document. |
| A2iA Locator  (multifield) | A2iA | OEM of <http://www.a2ia.com> Field Reader and Check Reader.  Uses dictionaries to read handwritten cursive text | Read checks, cursive handwriting, handwritten numbers. |

|  |  |  |  |
| --- | --- | --- | --- |
| Rules locators |  |  | Use Cases |
| Barcode Locator | BL | Find all 1D and 2D barcodes on a Document |  |
| Advanced Zone Locator  (multifield) | AZL | Read zones on a fixed-form document, with high control over OCR profiles and image cleanup. Advanced Registration  Cleanup Profiles  OCR Profiles  High precision  Slow to configure  Slow to run, Group Locators can be much faster. | Reading handwritten forms. |
| Format Locator | FL | Find all occurrences of regular expressions on documents.  Anchor management is rules-based | Find dates and amounts.  Warning: Not fuzzy – OCR errors will lead to missed data.  This locator runs very slowly if you have many regular expressions. |
| Invoice Header Locator  (multifield) | IHL | Consider deprecated. Use IGL & AGL instead.  Non-trainable locator for finding invoice fields | Find all invoice fields |

|  |  |  |  |
| --- | --- | --- | --- |
| Logic locators |  |  | Use Cases |
| Advanced Evaluator (formally Invoice Evaluator) | AE | Combine alternatives together from other locators and rescore them. | Choose best invoice date from 4 locators that attempted to find it. |
| Vendor Locator  (multifield) | VL | Recalculates the confidences of the alternatives of a database locator | Giving higher confidence to a vendor where the faxnumber is perfectly found on the document. |
| Standard Evaluator  (single field) | SE | Combines alternatives from two or more locators either as best of or first of. | Superseded by Advanced Evaluator, except that it can take more than 3 inputs.. |
| Relational Evaluator  (single field) | RE | Find an alternative of one locator that is left/right/above/below the alternative of another locator | Very rarely used. |
| OCR Voting  (multifield) | OV | Pick the advanced zone locator subfield with the best confidence |  |

# Training

If you understand well how the training mechanism in Kofax Transformation works, and you know how to train documents and configure the Advanced Evaluator (formerly the Invoice Evaluator) then you will be easily able to build projects that are fast to build and give the user a great experience in learning from their validation.

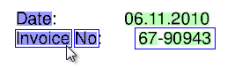
## How to Train Documents in Project Builder

Watch this video <https://www.youtube.com/watch?v=bIDmLVHmJNM#t=1035> to see how to train documents in Project Builder.

You can do all of the document training from within Project Builder. You do not (and should not) use KTM Validation to train your documents – you will be very slow, unable to improve the validation interface as you are working, debug problems, and your documents will provide poor field confidences at runtime.

## Selecting Anchors

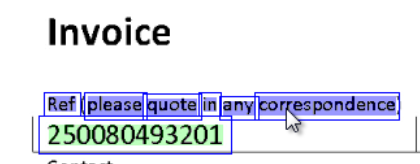
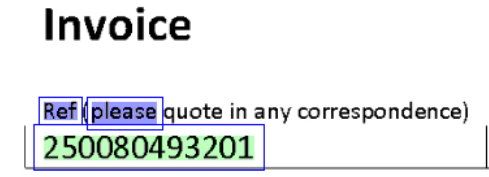
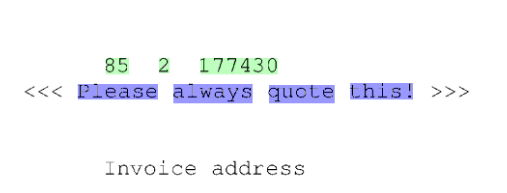
When a document is trained, the location on the page is stored along with any anchor words that are typically to the left or above the field



In the example above we see that the invoice number consists of 3 words “67”, “-“ and “90943”. The blue box surrounds the region on the page that KT will look for the invoice number. The two words to the left “Invoice” and “No” are marked as anchors. If you do not like an anchor, you **should change it**, by pressing **CTRL-RIGHT-CLICK** on the anchor.

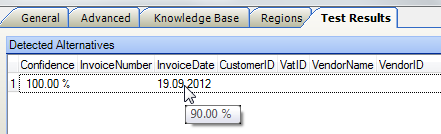
You should be spending a very long time checking and correcting anchors – it is a big part of the success of your project.

### How to Select Anchors

* You should pick anchors that do not have OCR errors if you can.
* Pick words that clearly identify the field for this particular document. The anchors can even be far away as “Total” is below.  
  
* Pick words that you know will help also on other documents of this class, as these words will go into the generic knowledge base to extract fields from unknown documents
* Do not pick anchors that are only coincidently there and can lead to confusion.
* In the example below KT picked too many anchors. Deselect what doesn’t make sense.  
   
* In the example below there was no “correct” anchor near to the invoice number. I chose to use “Please always quote this!” because it **will** help for this vendor and **shouldn’t** cause confusion on other documents.  
  

## Trainable Locator Subfield Confidences

Kofax Transformation **trusts** you as a user of Project Builder more than it trusts a Validation Operator when it comes to training documents. If you train a document in Project Builder, you only need **ONE** training sample to achieve a specific-training confidence of 90%. Pick a document with clear anchors and with long phrases in the fields.

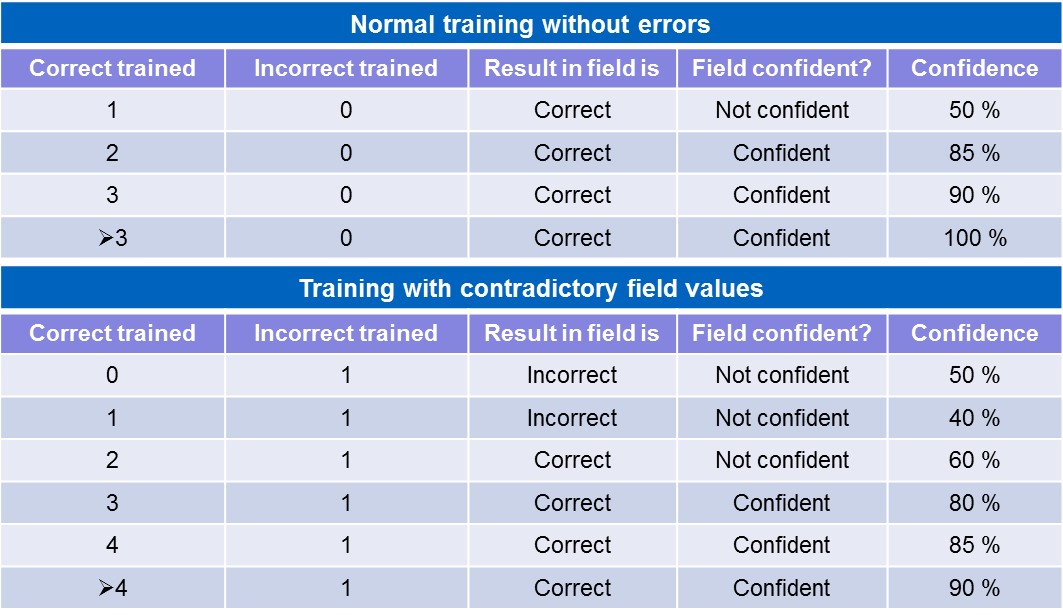


If a **second** documentis trained in Project Builder then the confidence is 95%. Documents that are trained in KTM Validation start out only with a 50% score.

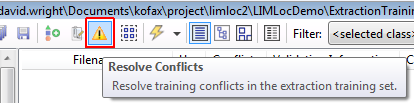
## Online Training Scoring and automatic conflict cleaning

If a document is trained in KTM Validation and then processed by KTM Learning Server, then the subfields in the next document will have confidence of 50%. After 2 documents the confidence will be 85%, 3 documents 90% and 40 documents 100%

If the **second** document conflicts with the first document because the field coordinates are different, then the confidence for the 3rd document is 40%, the 4th is 60%, 5th is 80%, 6th is 85% and 7th,8th, … stay at 90%. This is the conflict mechanism learning to ignore the conflicting documents.



## Resolving Conflicts

When you train your project with extraction samples, you can view the conflicts in the **Resolve Conflicts Screen**.  


Here you have the choice to fix the conflict or to delete one or both of the documents.

Conflicts occur when two documents with the same classification layout have the same field at different places on the screen.

## Invoice Evaluator/Advanced Evaluator

This is one of the most important locators in KTM, and is key to success in online learning.

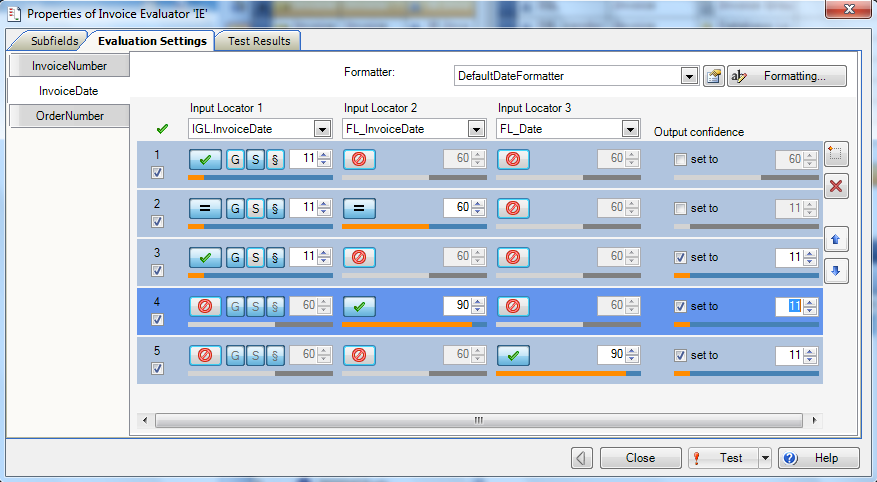
It was renamed to **Advanced Evaluator** in KTM 6.0 and is license-free. You should use it in probably any project.

The Invoice Evaluator requires an invoice add-on license in KTM5.5

You should only use this locator for **independent trainable fields,** e.g. invoice number, invoice date, order number, order date.

DO NOT put any amount fields into the IE, as they are dependent on each other and the Advanced Evaluator can mix unrelated alternatives

### Create a strong Learning Feedback Mechanism for the User

The most important principle of configuring IE/AE is to create a strong feedback mechanism for the user:

1. Make sure that what the user has trained always wins, no matter how bad the score.
   * This is strong feedback and makes the user realize, “KTM really learns from me. When I train well it does well, when I make a mistake it shows me my mistake again.”
   * That is why the first row is Specific with threshold of 11% (11% is just over the lowest threshold of 10% for a locator to be copied to a field).
   * Specific should always beat generic. Specific means “I have seen this document before”. Generic means “I haven’t seen this before but I think….”
   * Let the conflict resolution mechanism raise the scores of training files – you can set field thresholds less than 85% so that two samples give straight-through processing, greater than 85% for three samples, and greater than 90% for four samples.
2. Generic result agrees with a format locator that has evaluation words. – when non-determinism agrees with determinism.
3. Generic result is shown, but confidence set to 11% to require user validation.
4. Show the result from the format locator with evaluation words but give bad score.
5. Show the first result from the weaker format locator.

This is just a starting point for your Advanced Evaluator. We have configured it very aggressively above to avoid false positives. You may find that you can relax some of the tight thresholds by running and rerunning benchmarks with different configuration settings. You should never adjust these settings based on individual documents, but rather on larger groups of documents.

# Databases

KTM has an immensely powerful Fuzzy Database technology that should form the heart of many things that your project does.

Many KTM developers have a deterministic background and very familiar with SQL databases, but SQL databases are less than ideal in most KTM contexts because they are slow to query, require search-by-column and require perfect input, which you rarely have from OCR documents.

KTM fuzzy databases have the following advantages

* Queries are fuzzy, returning many alternatives
* Very fast. Even databases of 10’s of millions of records can be queried subsecond.
* It is easy to query the **entire text of a document** with a database. This is impossible with SQL.
* Users do not need to be aware of database columns
* They function like Google Search – any words can be used to search, and spelling mistakes are allowed. And usually what you are looking for is the first or second result.
* They are very interactive and easy to understand for the user
* They easily bring in data from other business systems making KTM a single cockpit for extracting data from documents.

## Database Locator

You should always consider using a Vendor Locator to rescore and filter the results of the Database Locator. Often it is easier to configure a vendor locator to clean up database locator results, than it is to configure a database locator to get the correct result every time.

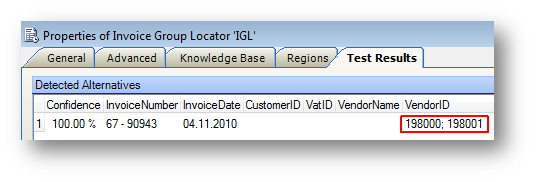
Vendor Names are often not the same on invoices as they are on documents.

Make sure you put city, street and post code in a group and restrict them to the same line.  


If the first alternative of the Exclusion Database has a match of 80% or more, then a dynamic exclusion zone is produced on the document. Use this to force the database locator to ignore certain results.

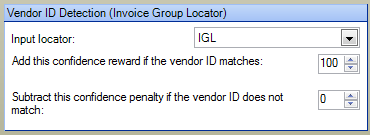
## Vendor Locator

This locator is important to be able to train VendorIDs, rescore the alternatives of the database locator and remove unwanted alternatives.

It also supports invoices with more than one Vendor ID. Perhaps one Vendor sends invoice to 3 departments and has 3 different VendorIDs in the database. The Vendor locator can resolve the correct Vendor. Below you see the Invoice Group Locator returning 2 trained VendorIDs for this invoice.  


Steps to configure the Vendor Locator

1. Make sure that the Database Locator is returning MANY results, not just 20.
2. Make sure that the Database Locator is giving scores of 40%-70% for most results.
3. Make sure that the Database Locator has the correct alternative in the top 20-50. It is not important for the database locator to get it right.
4. Use a Format Locator to find VAT ID candidates or Business Numbers or Fax Numbers, and configure the Vendor Locator to reward alternatives with 100% which match perfectly.
5. Reward the alternative that matches online learning with 100%.



1. Do not subtract confidences in the Vendor Locator. You do not want to punish a result because of an OCR error.
2. Always use the Extraction Benchmark to tune all of these options to find what works best with your documents and databases.

## Database Evaluator

The Database Evaluator is a seldom used Evaluator, but it has some wonderful use cases.

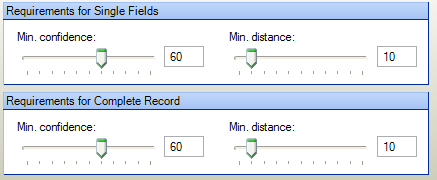
* Spell Checking
* Record Retrieval
* Fuzzy Webservice

### Spell Checking with Database Evaluator

You can use a Database Evaluator to spell check values from an Advanced Zone Locator.

If a zone has 100 or even 1000 possible values, put them into a fuzzy database with 1 column and use the Database Evaluator to automatically correct the spelling by selecting **Partial match required** and **Always allow.**

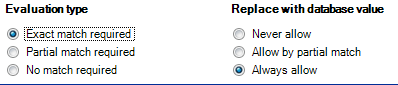


You can more finely control what happens by adjusting the sliders.  


### “Webservice” or “SQL” query with Database Evaluator

You can use the **Search & Matching Server** and a **Database Evaluator** with **Exact match required** to have a web-service like call, complete script-free, instead of writing a webservice to access a database and then KTM scripting to call the webservice,

Choose “partial match required” to have a “fuzzy” webservice call.



## Validation Form Search

You might have a field for “US State” on a document. You could create the following database from <http://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations>

|  |  |  |  |
| --- | --- | --- | --- |
| State | USPS | Other | ANSI |
| Alabama | AL | ALA | 01 |
| Arizona | AZ | ARI | 02 |
| Alaska | AK | Alas | 02 |
| … | … | … |  |

You could use the above US State database in a field formatter to spellcheck words. If the user types “AZ” and ENTER, the text will be replaced with “”Arizona”. They could also type Arizo or Abalama or 01 as well. See Calling a Fuzzy Database in a Field Formatter for Spell Checking

Now you have a very flexible, fuzzy and fast way of entering US States, and each user can use whatever system they prefer.

# Search And Matching Server

## Kofax Search & Matching Server KSMS has the following value

* Security

o Customer databases need to be secure and encrypted. It should not be possible for anyone to steal or leach data from it.

o The Customer’s data stays in the data center, not in the application center.

o The KTM developer should have zero access and no read rights to customer databases.

* The project has >>100 MB of fuzzy databases

o These should not be loaded into every Validation Client into RAM because it is slow to open module, uses RAM, and is a security risk.

o These should not be loaded into every KTM Server CPU instance, because 16 CPU times lots of MB = lots of GB of RAM wasted.

* Integration Effort

o The customer doesn’t need to write a tool to push the data regularly to KTM, because KSMS pulls the data.

* KSMS bridges the deterministic world of SQL and web programming with the Quantum, fuzzy world of KTM, and makes a lot of deterministic and anachronistic webservice and dll integrations to KTM unnecessary.
* KSMS should be sold to the DB Admin as a free and simple-to-install-and-configure fuzzy indexer for any SQL database. This fuzzy database/view is something that they control and can completely lock down.

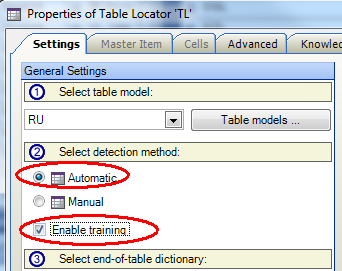
## Comparison between different Database Types

You should try to get ALL data as fuzzy. Avoid SQL and webservices if you can.

Determine a data strategy considering advantages of SQL, webservices, FUZZY Databases, Search&Matching server, Flat Files or XML.

|  |  |  |
| --- | --- | --- |
| Database Type | Advantages | Disadvantages |
| Fuzzy Local | Fast  Zero Latency  Immune to OCR errors  KTM has no access to Enterprise Database Infrastructure  Requires no scripting  Also supports non-fuzzy searches | Read only  Entire database & index transferred to clients as plain text  Database Admins need to generate csv files periodically. |
| Fuzzy Remote  (Search & Matching Server) | All above  Secure and scalable.  KTM has NO access to Enterprise Database Infrastructure  Supports 10s of millions of records  Converts any SQL database into a fully indexed fuzzy database  No need for Validation clients to connect directly to databases.  Easy for DB Admin to use.  License Free. | Higher latency than local fuzzy |
| SQL | KTM scripting supports “SELECT”.  . | Fail on any OCR errors.  Latency problems  Security risk. ODBC config and or passwords in plain text on clients.  Many users have access to data.  Requires complex scripting  Need to use ADO.NET or ODBC to do INSERT. |
| Webservices | Well known | Deterministic  Fail on any OCR errors  Can have latency problems  DLLs need installing on all clients |
| Dictionaries | Easy to create | Require project republishing |

# Tables

Line Item matching is the most time-consuming task for account processing clerks, and it can easily provide the largest return on investment in a KTM solution.  


KTM’s Table Locators use at least seven different methods to find the line items of an invoice and can be difficult to configure optimally.

We recommend combining Manual and Automatic Methods. This is why trained tables use the Manual Method and unknown tables use the automatic method.

## Automatic Table Locators

The automatic table locator is used to locate the table on an unknown invoice. It attempts to find:

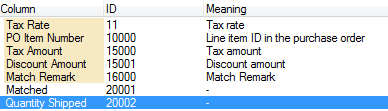
1. The table header
2. The rows and the columns
3. The identity of the columns
4. The end of the table

KTM 5.5 added the ability to the Table Locator to take the Line Item Matching Locator’s table as input.

To detect the header of a table, you will need to train 50-100 invoices in Table header packs per language for this to be successful.

### Table Model

Make your table model as simple as possible. Your table model should exactly the columns required for data entry into the ERP system. But use the built-in KTM columns where possible, because they are used by the detection algorithms. Custom columns will require header detection to be found.

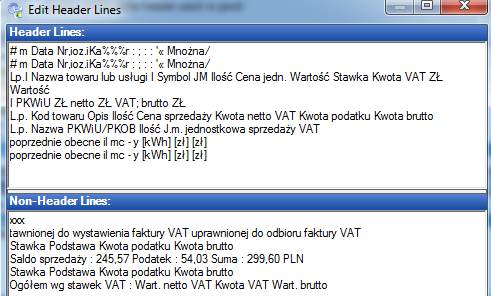


Note: the “Unit Measure” column looks for scaling units: 1, 100, or 1000. It not kg,m², and so on.

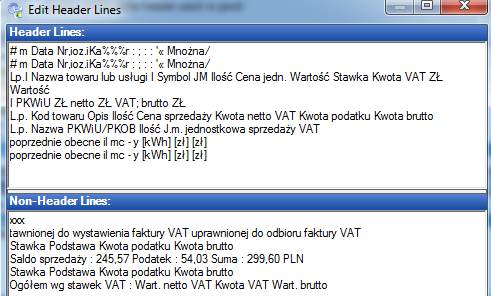
### Table Header Pack

A header pack consists simply of three text files. Try to make these files as complete as possible. Correct OCR errors manually. Header detection works best on column names that have no word-wrapping.

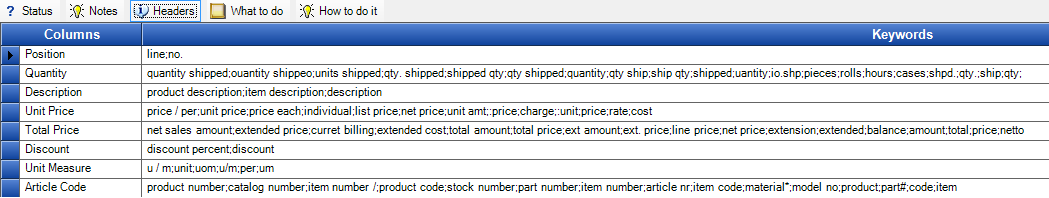
1. Header row examples:



1. Non-Header row examples:

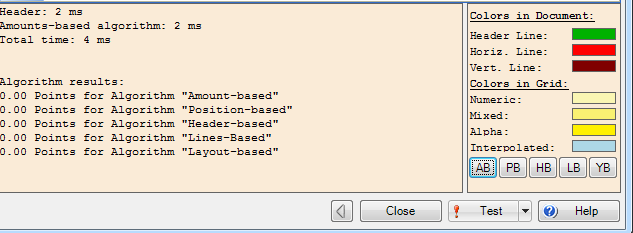


1. Header Keywords



### Row and Column Detection

The next step applies five different algorithms that attempt to read the lines under the header and interpret the columns. To switch the Table Locator configuration panel into Debug Mode where you can test individual algorithms, press Ctrl-F12. The algorithm with the best score wins – the other four are ignored.



### Amount-based Algorithm

Attempts to find mathematical relationships in rows (e.g. A\*B=C).

To improve it, make sure that your format locator that feeds the table locator is finding the amounts in the table.

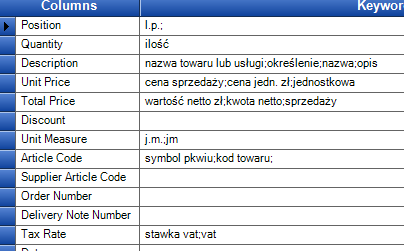
### Position-based Algorithm

Attempts to find an increasing number, probably on the left of the document, which represents the position number in the invoice. Ideally it is 1, 2, 3, 4, 5, 6, …..

### Header-based Algorithm

Attempts to use the header words to understand where the columns are.

To improve it, make sure that your Table Header Pack is complete with all possible keywords for the columns. This example would fail to find the discount and supplier article code columns. There are only four description column names, which are also probably too few. Correct the spelling of the keywords to improve fuzzy scoring across all of your invoices.



### Line-based Algorithm

Attempts to find the table cells based on horizontal and vertical lines that are on the pages. You cannot improve how this algorithm works.

### Layout-based Algorithm

Attempts to use white space between columns to understand where the columns are. You cannot improve how this algorithm works.

## Manual Table Locator

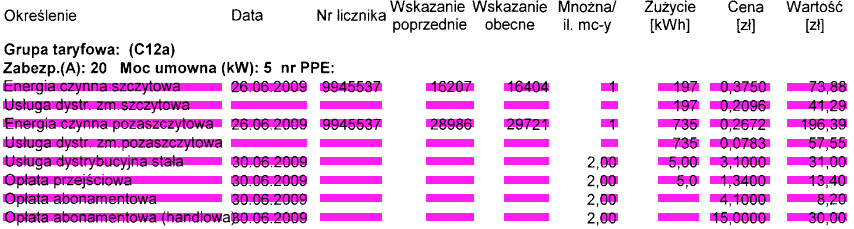
This second mode of the Table Locator uses sample documents to identify a previously-seen table.

You can either make subclasses for important vendors with their own Manual Table Locator, or use the Online Learning Mechanism’s specific knowledge bases to store these sample tables.

The easiest way to understand the table locator is to imagine a stencil (<http://en.wikipedia.org/wiki/Stencil>) being moved down the page looking for matches. It does not look for the table header.

**Tips**

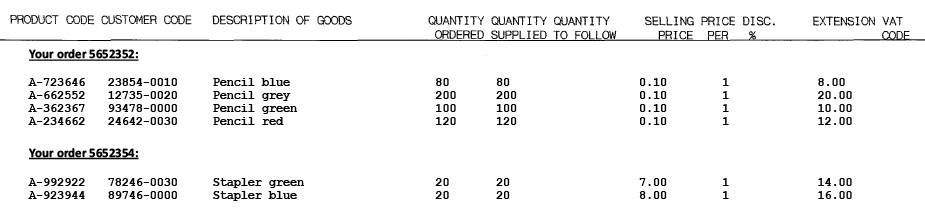
* The “master item” should be a table row containing optional row and long descriptions.
* The “master item” should not have missing cells.
* Mark the optional rows in the master item.
* Expand the left and right edges of the cells.
* Make sure that you mark optional cells.
* Cell Types are internal algorithms that attempt to work it if the cells are numeric or alphabetic.
* After online learning, check and correct specific training samples in the Edit Document Dialog (press F10 on the Xdoc).
* Anchors are words or phrases that appear on each table row, but are not needed to be read – they help with matching the stencil.
* In the example below we see that columns 2-6 are optional, and that column 6 s too narrow for the “2,00”. However the 2,00 will be correctly detected. But if the OCR engine sees “2 00” then only 00 will be found.
* The last row will have “(handlowa)80.06.2009” in column 1 because the characters run together.



Both manual and automatic table detection can search for interleaved and embedded columns.

### Interleaved Columns

An interleaved column is a value that applies to all the rows below it, like order numbers in the example below that appear before the first and fifth line items.



### Embedded Columns

An embedded column is typically an article number or date appended to a description.

## Line Item Matching Locator

LIMLoc matches the words on the invoices with the open purchase orders from the ERP system. It requires an SQL connection to a database containing open purchase orders.

If LIMLoc does not have a vendor ID, it cannot work.

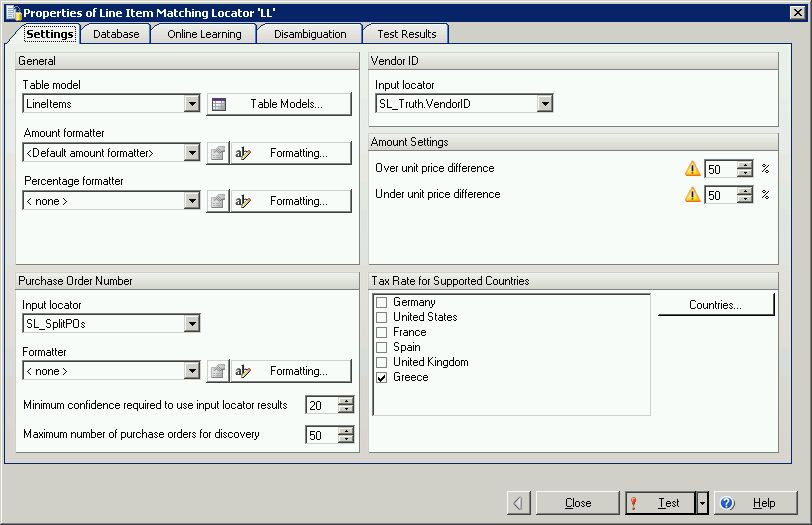
If no PO numbers are provided, LIMLoc matches against the first 20 open orders in the database.

LIMLoc has a self-contained learning mechanism to improve its extraction.

LIMLoc returns tolerance messages for each line item.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RemarkID | Valid | Message |  | Name |
| 1 | True | OK |  | IDS\_MatchRemark\_Match |
| 2 | True | Under Or over delivery |  | IDS\_MatchRemark\_UnderOverDelivery |
| 3 | True | Change In units of measurement |  | IDS\_MatchRemark\_UomChanges |
| 4 | True | Under Or over price |  | IDS\_MatchRemark\_UnderOverPrice |
| 5 | False | Missing unit price |  | IDS\_MatchRemark\_MissingUnitPrice |
| 6 | True | Unit price scaled by power of ten |  | IDS\_MatchRemark\_ScaledUnitPrice |
| 7 | True | Total amount scaled by power of ten |  | IDS\_MatchRemark\_ScaledTotalAmount |
| 8 | False | OCR errors |  | IDS\_MatchRemark\_OcrErrors |
| 9 | False | Only total amount Matched |  | IDS\_MatchRemark\_OnlyTotalAmount |
| 10 | False | Several matches possible |  | IDS\_MatchRemark\_Ambiguous |
| 11 | False | Only quantity, unit price and total amount Matched |  | IDS\_MatchRemark\_UnmatchedConsistent |
| 12 | False | Could Not Match |  | IDS\_MatchRemark\_Unmatched |
| 13 | True | Validated by user |  | IDS\_MatchRemark\_UserValidated |

Notes to Line Item Matching Locator

* remarkID 6 and 7 only look for scaling of 10, 100 and 1000.
* Multiple PO numbers can be passed in as either
  + a semicolon separated value
  + a list of alternatives
  + a list of subfields.
* Limloc also checks at PO numbers that are not passed in as parameters, meaning that matches can still be found if a PO number is not found from multiple POs.
* Over & Under Delivery has a tolerance of +-3%. This can be extended much further with KTM 6sp1.  
  
* If there are discounts in the line item the Unit Price is replaced with the discounted amount.
* Limloc only supports tax discounts that are the same on all line items.

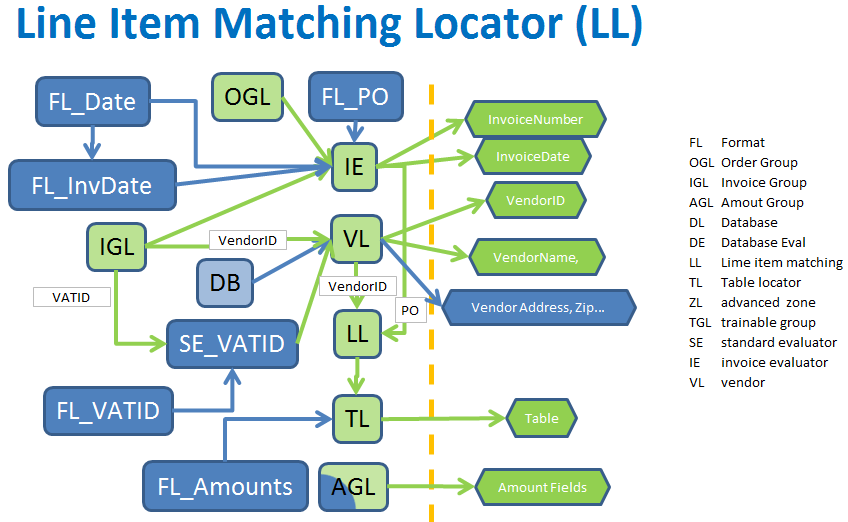
## Combining Locators

You will obtain the best results with table detection by combining together these ten locators.

|  |  |  |
| --- | --- | --- |
| Locator Name | Locator Type | Function |
| IGL | Invoice Group | Finds the VendorID from Online Learning |
| OGL | Order Group | Finds the Order Number from Online Learning |
| FL\_PO | Format | Finds PO Number candidates |
| IE | Invoice Evaluator | Combines OGL.PO and FL\_PO |
| DB\_Vendor | Database | Finds vendor candidates |
| FL\_VATID | Format | Finds VAT ID candidates |
| VL\_Vendor | Vendor | Uses IGL.VendorID, DB\_Vendor and FL\_VATID to find the Vendor |
| LL | Line Item Matching | Uses IE.PO, VL\_Vendor.VendorID and the SQL access to the Open PO database to find the line Items |
| FL\_Amounts | Format | Finds any amounts on the document |
| TL | Table Locator | Uses LL and FL\_Amounts to find the table. |

The diagram below shows how to connect all the locators together for the Line Item Matching Locator.

**Green** marks learnable fields, **Blue** are not learnable.



## Table Interpolation in Validation

You can use the table interpolation feature in KTM Validation to quickly extract tables and also to train tables. You will need to examine these specific training samples in Project Builder by pressing F10 on the Xdocs.

The Interpolation Button can be pressed more than once. Interpolate a row, select another row and then interpolate that. This will successively find more of the table.

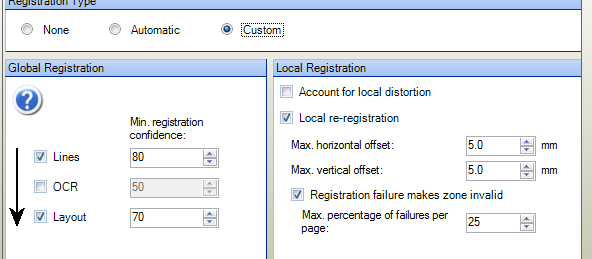
# Advanced Zone Locator

You should use the inbuilt registration technologies before attempting to build anchors. Configuring registration is easy and faster to run and test.

Anchors are slow and tedious to configure, only use them if registration fails.

## How to configure Registration

The default settings for registration are very strict and can often lead to registration failure. This is one reason many go straight to anchors. Loosen and tune these registration settings for better results.

* Make a benchmark set and choose settings that work across all documents, not individual documents.
* Try tuning automatic registration before attempting to make anchors.  
  
  + Select custom registration.
  + Try layout first, then lines and then text. Lower thresholds.
  + Deselect “registration failure makes zone invalid”, so you can see the registration results.
  + Try turning on “account for local distortion”
  + Increase offsets.

## Anchors

### Text Anchors

Make sure your OCR Profile is multiline.

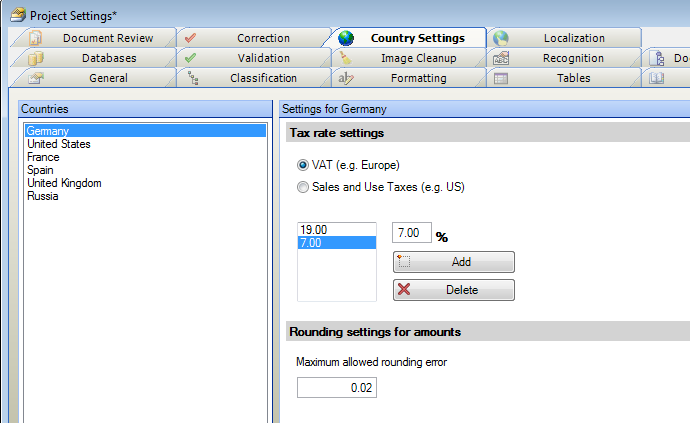
### Structural Anchors

Make sure that line length and thickness is appropriate for your documents

# Localization

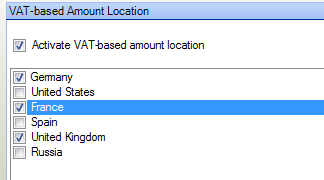
There are different kinds of localization in KTM

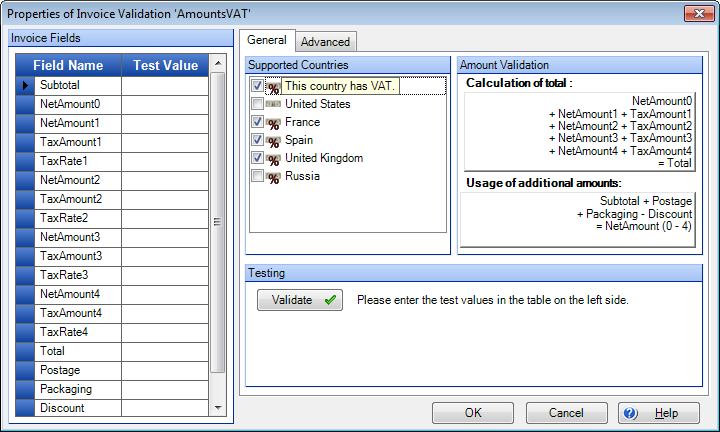
* Product - 10 Kofax Languages
* Project - ProjectSettings/Localization. Hierarchical Field/ErrorMsg localization mechanism using XML.
* Locator - Country Settings, dictionaries, validation rules......
* Localize Locators in KTM



Activate VAT countries in Amount Group Locator for Value Added Tax or GST (Goods and Services Tax)

You cannot mix SalesTax and VAT/GST into one locator.

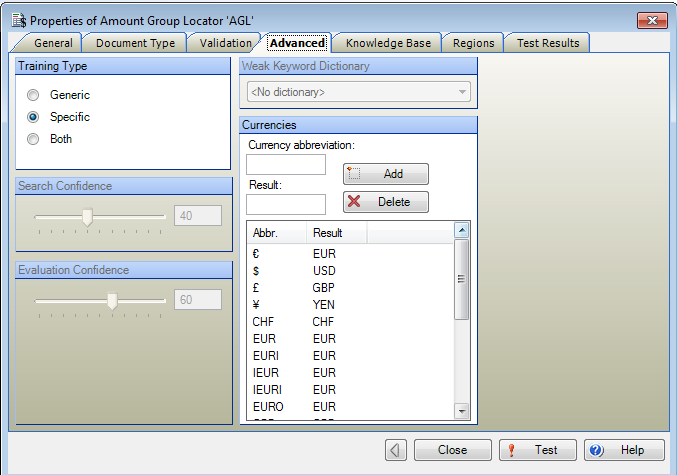


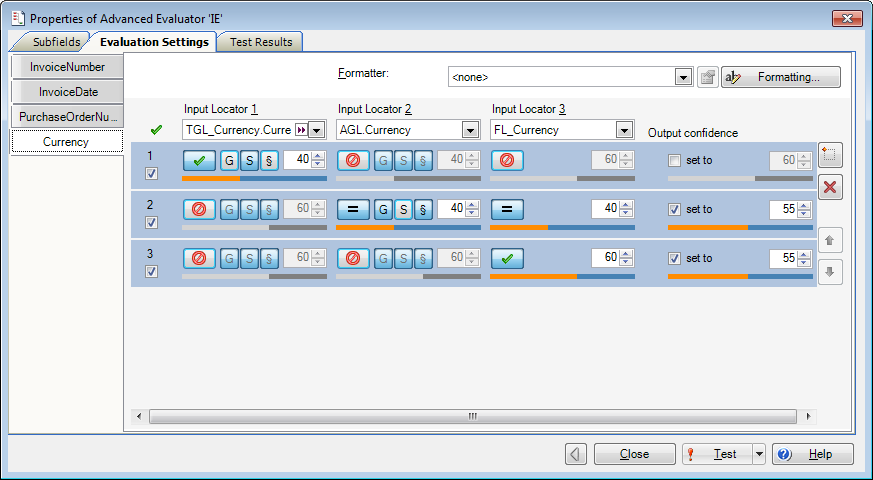
Select valid VAT rates in the Invoice Validation Rule.  


**Invoice Group Locator/AmountGroupLocator/OrderGroupLocator**

Load your knowledge bases (InvoiceAddOnPack comes with en\_us, en\_uk, De,ES,Fr)

Amount Group Locator has a hardcoded list of currency expressions

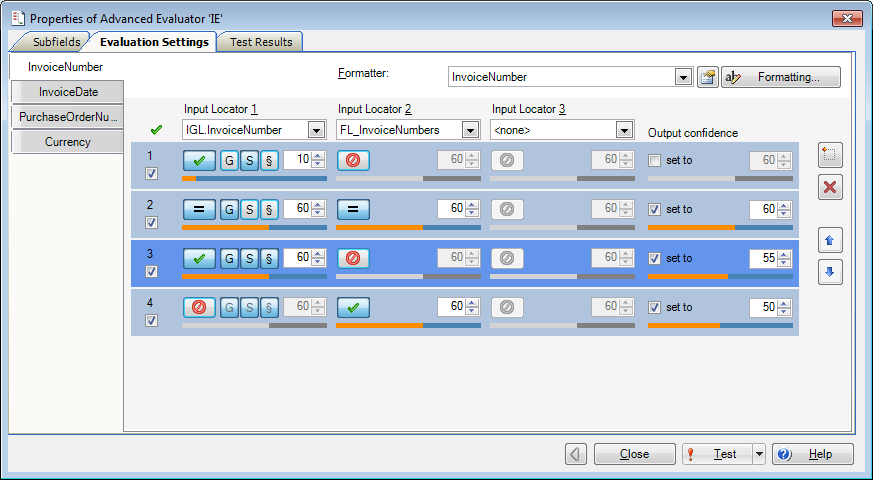


The InvoiceEvaluator.Currency is favouring the Trainable group locator over AGL for currency detection (which is a good idea)  


InvoiceEvaluator.InvoiceNumber

It is CRITICALLY important that the IGL ALWAYS beats the FL in invoice number and invoice date detection, because the learning system MUST ALWAYS beat hardcode dictionary lists. Dictionary lists are only backup mechanisms – they can never override human corrections.

In an untuned project we always start with high mistrust of FL’ dictionaries. This can be relaxed based on large benchmarks.

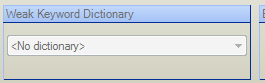


## Dictionaries

There are dictionary files in the invoice add on pack. You can add more words. Focus on putting a few good strong phrases into your dictionaries

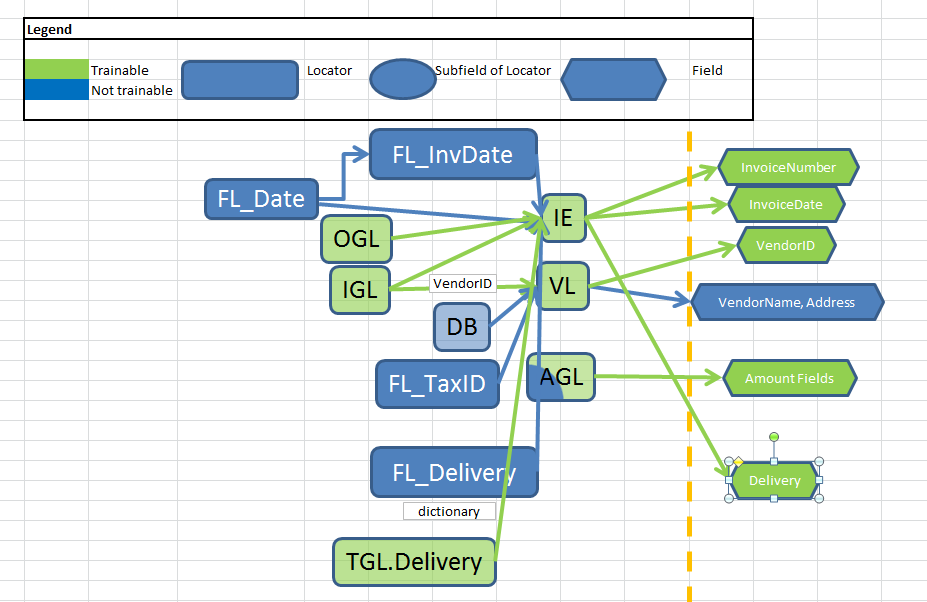
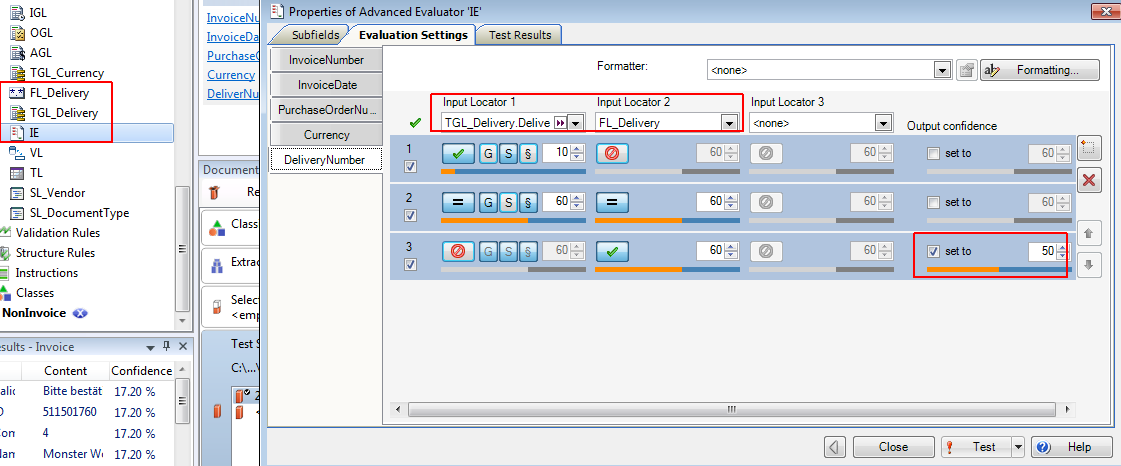
**STRONG PHRASES – not ambiguous.** These words ALWAYS mean your field “invoice number”, “invoice num”

**WEAK PHRASES – ambiguous,** “Number”**,** “invoice” **,**”Reference”

The Invoice Group Locator (IGL) and Order Group Locator (OGL) have weak keyword dictionaries, which tell the locators what words are weak (i.e. ambiguous). These allow the OGL & OGL to **reduce** the confidence of ambiguous results.

**Delivery Number strategy**

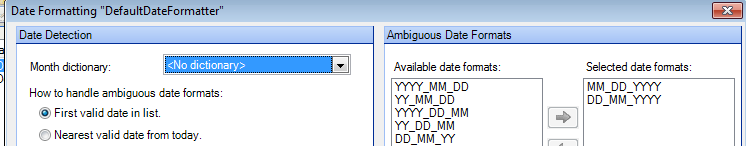


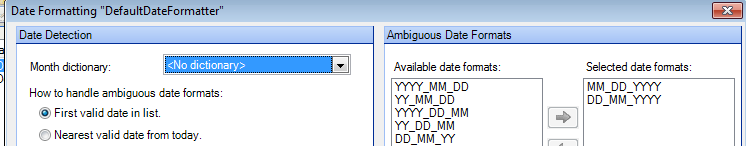


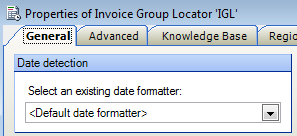
In KTM 6 the “Invoice Evaluator” was renamed to “Advanced Evaluator”

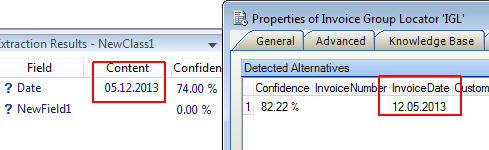
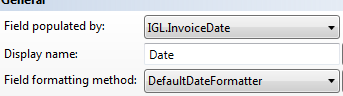
## Dates

You cannot mix US and European invoices.

If you configure a Field Formatter to support both DD\_MM\_YYYY and MM\_DD\_YYYY you will end up with flipping numbers.  




IGL uses the field formatter.  


Field uses the field formatter.  


Validation GUI uses the field formatter when you press ENTER





# Batch Restructuring

In the events **Batch\_Open** and **Batch\_Close** you have access to the global object **Batch**, which contains many straightforward methods for moving, deleting and copying documents and pages and also for creating child batches.

## Workflow Agents not needed

Workflow Agents are used in Kofax Capture to split batches, route documents, report statistics, change user permissions and other things.

Most of these tasks are far easier to implement in KTM than to write a DLL for Kofax Capture.

In the **Batch\_Close** event you can split, merge, move and delete documents, create child batches, reject documents and pages and skip modules.

You cannot however change batch permissions

|  |
| --- |
| Private Sub Batch\_Close(ByVal pXRootFolder As CASCADELib.CscXFolder, ByVal CloseMode As CASCADELib.CscBatchCloseMode)  Batch.DeleteDocument(pXRootFolder,3)  If pXRootFolder.Valid And Project.ScriptExecutionMode=CscScriptExecutionMode.CscScriptModeServer Then  pXRootFolder.XValues.Set("KTM\_DOCUMENTROUTING\_QUEUE\_THISBATCH", "KC.PDF")  End If  Batch.MoveDocumentTo(pXFolder,0,NewFolder,1)  …. |

The following read/write XValues can be used to reject documents and pages

|  |
| --- |
| XDocInfo.XValues.ItemByName("AC\_REJECTED\_DOCUMENT")  XDocInfo.XValues.ItemByName("AC\_REJECTED\_DOCUMENT\_NOTE").  XDocInfo.XValues.ItemByName("AC\_REJECTED\_PAGE2") 'reject the 3rd page  XDocInfo.XValues.ItemByName("AC\_REJECTED\_PAGE\_NOTE3") |

## Splitting Batches

|  |
| --- |
| 'Split a document to another batch  pXRootFolder.DocInfos(x).XValues.Set("KTM\_DOCUMENTROUTING", ChildBatchName)  'Route a child batch to another KC module  pXRootFolder.XValues.Set("KTM\_DOCUMENTROUTING\_QUEUE\_" & ChildBatchName, "KC.PDF")  'Rename a child batch  'If this step is skipped, then the batch will be automatically named  pXRootFolder.XValues.Set("KTM\_DOCUMENTROUTING\_BATCHNAME\_" & ChildBatchName, pXRootFolder.XValues.Item("AC\_BATCH\_NAME") & " " & ChildBatchName) |

# Auditing a KTM Project in production

1. Benchmark Data
2. Observe Users using KTM
3. Observe processes before and after KTM
4. Project Examination
   1. Examine Project Class Tree
      1. Good hierarchy
      2. Field and locator inheritance
   2. Field Names (consistent, do they all come from locators?)
   3. Locators
      1. (order, naming FL\_, SL\_, AGL\_, DL, etc)
   4. Field Formatters
      1. Well named, simple. Are they used?
   5. Validation Rules
      1. Every editable field must have at least one validation rule.
      2. Use multifield validation rules even for single fields, so you can access pXDoc
   6. Validation Forms
      1. Does it have good usability? “Mouse-lessness”, cursor, keyboard shortcuts, eye movement?
   7. Run the benchmarks
      1. Start fixing false positives
      2. Reduce false negatives
      3. Reduce true negatives.

# Scripting Guidelines

Many KTM projects have thousands of lines of complicated scripting. If you follow these principles below, you will have that scripts that are easy to test, debug and understand, short, reusable in other projects. If your scripts are longer than 1000 lines then they are probably too complicated.

## Productivity, before Accuracy

Spend more time writing scripts for user productivity, than in writing scripts to improve OCR results or reduce true negatives. Don’t be distracted by clever ideas that may not bring much value to the user

## Divide & Conquer

“Find & Filter” -It’s often much easier to find alternatives and then filter the bad ones rather than build a complex logic to find only the good ones. Let a table locator find too many rows, and then write a script locator to delete the unwanted rows.

“Find & Fill” – one locator may not find all values, use a script locator to find the rest, such as missing table rows or cells

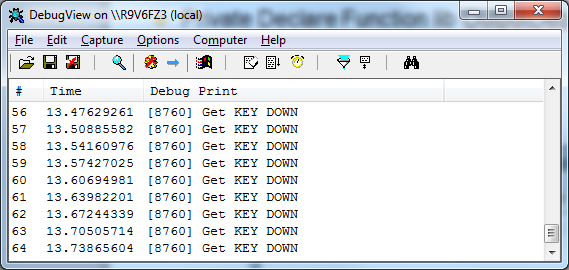
## Debugging

Debugging in Project builder

* Use SysInternals’ Debugview to view, filter and log to file safely in   
  a multithreaded 64 bit environment.
* On a 64 bit machine later than Vista you will have to set this registry entry  
  HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Debug Print Filter

Add the following value DEFAULT : REG\_DWORD : 0xFFFFFFFF

* You may also know Sysinternals’ other useful products TCPView and Process Explorer  
  <http://technet.microsoft.com/en-us/sysinternals/bb896647.aspx>



## Error handling

Do not ever use “On Error Resume.” – you will miss unusual errors and there will be no messages.

Do not use “on error goto” unless:

* There is no other way
* You have only one line of code to test

Use Err.Raise to throw errors HARD. This will reject the batch and force it to Quality Control with an easy-to-read message both in QC and Batch Manager. In Project Builder the debugger will open with a breakpoint on the error.

|  |
| --- |
| Then Err.Raise 34588,,"Column '" & columnName & "' does not exist in database '" & dbname & "'." |

It is better that an error goes to Quality Control and is visible and fixed, than landing in an unread log file, which may take days to find and understand

Run a Benchmark with 100’s or 1000’s of files to fix your bugs before you go into production.

## Separate Data from Logic.

Use Built-in script variables and script resources or external xml, txt, databases for parameters instead of constants and global parameters. You will not be able to fix them without republishing – and your customer will not be able to manage their data without your involvement.

Your code will be smaller, more readable, portable, and your customers can fix data problems themselves.

## Always assume data is bad, wrong or missing

Validate everything. Check boundary conditions – never assume something exists. Check that arrays or Alternatives are not empty.

## Put scripts in the right events

Scripts generally do one of the following - Classify, locate, , format, validate.

|  |  |  |
| --- | --- | --- |
| Script Task | Where to put it | Comment |
| Add/Remove/Delete pages or documents | **Batch\_Open** & **Batch\_Close** | Do everything with the global object Batch, so your results are synchronized to Kofax Capture |
| Classify Documents | **Document\_AfterClassifyXDoc** | Use pXDocument.Reclassify(name, confidence) |
| Find Data on a Document or use other locators | **Script Locator** |  |
| Change the Format of a field value | **Field Formatter** | These run after extraction in Server, and after the ENTER key is pressed in Validation |
| Check if a field value makes sense | **Validation Method** | These run after fields are formatted in the Server and Validation Modules. |
| Button Click in Validation | **ValidationForm\_ButtonClicked** | Do not try to format or validate data here. Simply carry out the action of the Button which should only be inserting data into the field. |

Do not use **Document\_AfterExtract** to fill fields, use Script Locators

Use **Document\_AfterExtract** only for these unusual cases

* Except for nested classes where you would need to duplicate script locator code.
* for formatting Zonal Reread values.

Use Formatters and Validation rules, not GUI Events in Validation Form.

## Use well defined function names

Name all of your functions and subroutines Object\_Method. This helps you organize your scripts better and make them more readable.

## Refactor continuously.

Don‘t duplicate with copy/paste. Write functions and reuse them. Isolate your functions by only passing them the parameters they need – only very rarely do you need global variables in KTM – and they may be destroyed if a user clicks to a previous document, or if KTM server is processing documents in parallel.

Sample functions for Naming and Isolation. (These functions are very reusable in other projects)

|  |
| --- |
| Private Function String\_CountDigits(a As String) As Long  Public Function String\_RemoveAllSpaces(a As String) As String  Private Function Table\_SumColumn(table As CscXDocTable, colID As Long,amountFormatter As ICscFieldFormatter,ByRef sum As Double) As Boolean  Public Function XDocument\_FindLeftTextMargin(pXDoc As CscXDocument,p As Long) As Long  Public Function XDocument\_GetNextPhrase(ByVal pXDoc As CscXDocument,subfield As CscXDocSubField, pixels As Long) As CscXDocWords  Private Sub Batch\_SplitIntoSmallerBatches(ByVal Folder As CscXFolder, batchSize As Integer) |

## Use KTM‘s Object Model. This will simplify your code and make it easier to read. Avoid using Strings.

The following example of a script locator uses many local variables, and none of them strings.

Private Sub SL\_FillTable\_LocateAlternatives(ByVal pXDoc As CASCADELib.CscXDocument, ByVal pLocator As CASCADELib.CscXDocField)

Dim l As Long, tablestart As Long, cols As CscXDocFieldAlternatives, a As Long, c As Long, words() As CscXDocWords, w As Long, row As CscXDocTableRow, table As CscXDocTable

Dim textline As CscXDocTextLine, newrow As Boolean, prevRow As CscXDocTableRow

Set textline=pXDoc.TextLines(l)

Set words(c)=pXDoc.GetWordsInRect(textline.PageIndex,cols(a).Left,textline.Top,cols(a).Width,textline.Height)

For w = 0 To words(c).Count-1

If words(w).LineIndex=l Then row.Cells(c).AddWordData(words(w))

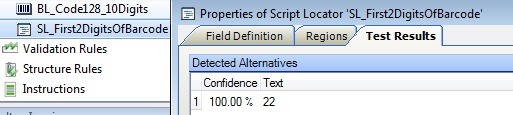
Next

Use CSCXDocWord(s), CSCXDocField, CSCDatabase, CSCXDocFieldAlternative(s), CSCXDocSubfield, CSCXDocTable

Instead of Strings. Your code will shorter and easier to read. Your will also discover more powerful things that you can do

## How to write a Script Locator with a single Subfield

The following example takes the first two digits from a barcode.



|  |
| --- |
| Private Sub SL\_First2CharactersOfBarcode\_LocateAlternatives(ByVal pXDoc As CASCADELib.CscXDocument, ByVal pLocator As CASCADELib.CscXDocField)  Dim Barcodes As CscXDocFieldAlternatives  Set Barcodes=pXDoc.Locators.ItemByName("BL").Alternatives  If Barcodes.Count=0 Then Exit Sub ' No barcode was found, so we cannot classify  If Len(Barcodes(0).Text)<2 Then Exit Sub 'This is not a valid barcode for classification  'Create an output alternative for this script locator  With pLocator.Alternatives.Create  .Text=Left(Barcodes(0).Text,2)  .Confidence=Barcodes(0).Confidence 'Copy the confidence from the barcode locator (it will always be 100%)  End With  End Sub |

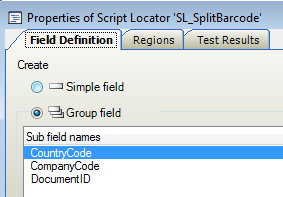
## How to write a Script Locator with multiple Subfields

Let’s say a barcode needs splitting into a 2 digit Country Code, 3 digit Company Code and 9 digit DocumentID.

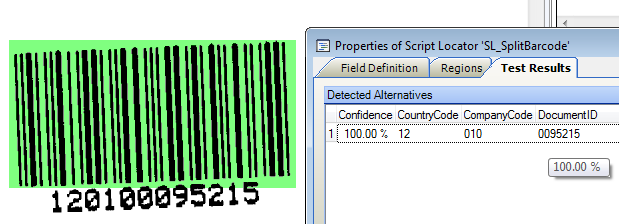
We have a Script Locator following the Barcode Locator



Give the Script Locator 3 Subfields.



Below you can see the output of the script locator including alternative confidence in the first column and subfield confidence in the mouse hover text.



|  |
| --- |
| Private Sub SL\_SplitBarcode\_LocateAlternatives(ByVal pXDoc As CASCADELib.CscXDocument, ByVal pLocator As CASCADELib.CscXDocField)  'Split a barcode into 3 pieces  Dim barcode As CscXDocFieldAlternative, alternative As CscXDocFieldAlternative, subfield As CscXDocSubField  With pXDoc.Locators.ItemByName("BL").Alternatives  If .Count=0 Then Exit Sub 'The barcode locator failed to find a barcode. Exit!  Set barcode=.ItemByIndex(0) 'Get the first barcode found by the barcode locator  End With  If Len(barcode.Text)<>12 Then Exit Sub ' this is the wrong barcode!!  'We now know we have a 12 digit barcode to split, so create the alternative to hold the results of the script locator  Set alternative=pLocator.Alternatives.Create  alternative.Confidence=barcode.Confidence 'Copy the barcodes' confidence  Set subfield=alternative.SubFields.Create("CountryCode")  subfield.Text=Left(barcode,2)  subfield.Confidence=barcode.Confidence  Set subfield=alternative.SubFields.Create("CompanyCode")  subfield.Text=Mid(barcode,3,3) ' the 3rd, 4th and 5th characters of the barcode  subfield.Confidence=barcode.Confidence  Set subfield=alternative.SubFields.Create("DocumentID")  subfield.Text=Mid(barcode,6) '6th character and following  subfield.Confidence=barcode.Confidence  End Su |

## “In-Place” Locator Alternative Editing

In-place editing is a technique you can use to change the alternatives of preceding locators directly.

KTM runs all of the locators and afterwards copies locators into fields. This means if a locator changes a previous locator’s results then the changed results will ultimately be copied to the fields.

Using in-place editing is better than writing code in Document\_AfterExtract event, because you are still using all of the field formatters, validation rules, and field thresholds and confidences correctly. You also do not interfere with the learning mechanisms.

The only disadvantage to using “in-place” editing is that you cannot see the results of your script locator when you press TEST. Also script locators don’t have the option to run all previous locators when testing. I usually put a breakpoint (F9) in my script and then process (F7) on a document to test and show the variables in the Watch Window (CTRL-W).

There are 3 occasions when this is important.

### Multifield Script Locators

Let’s say that you want to change the results of the Amount Group Locator, which has 24 subfields.

Instead of creating the exact same 24 subfields in the script locator and copying all the values 24 times (and losing trainability for the fields) just do the following.

|  |
| --- |
| Private Sub SL\_ChangeAGLResults(ByVal pXDoc As CASCADELib.CscXDocument, ByVal pLocator As CASCADELib.CscXDocField)  Dim alts As CscXDocFieldAlternatives, a As Long  Set alts=pXDoc.Locators.ItemByName("AGL").Alternatives  'If a currency is not found, set it to USD  For a = 0 To alts.Count-1  With alts(a).SubFields.ItemByName("Currency")  If .Text="" Then .Text="USD": .Confidence=1.0  End With  Next  End Sub |

### Manipulating Trainable Locator Results

Script locators do not produce trainable results. So either use in-place editing on the trainable locator, or use an Evaluator to combine results together in the way you like.

### Scripting Tables

Script locators do not support table objects. Table Locators are trainable. By editing or creating table rows with “in place” scripting, you can have automatic table location, specific knowledge bases, table interpolation in Validation Module and table learning all working together.

A Table Locator **always** creates a table object, even when no rows are detected.

The only way to give table cells coordinates is to use cell.AddWordData(word). Use either pXDoc.GetWordsInRect() or Dim Word as new CSCXDocument to make words.

If you delete rows from a table, make sure that you handle the row index carefully – this is easiest by starting at the last row of the table.

If specific training found a table that you want your script to ignore then use the following line

If pXDoc.Locators.ItemByName("TL").Alternatives(0).Source= \_ CscXDocAlternativeSource.CscASSpecific Then Exit Sub

Example of “in-place” editing of a table.

|  |
| --- |
| Private Sub SL\_CopyEmptyCellsFromAbove\_LocateAlternatives( \_  ByVal pXDoc As CASCADELib.CscXDocument, \_  ByVal pLocator As CASCADELib.CscXDocField)  'A table Locator's Table is stored in its first alternative  Dim table As CscXDocTable  Set table=pXDoc.Locators.ItemByName("TL").Alternatives(0).Table  Table\_CopyEmptyCellsFromAbove(table,pXDoc)  End Sub  Public Sub Table\_CopyEmptyCellsFromAbove(table As CscXDocTable,pXDoc As CscXDocument)  'Copies the values of cells to empty cells below  Dim row As CscXDocTableRow,r As Long  Dim cellAbove As CscXDocTableCell, cell As CscXDocTableCell, c As Long  Dim words As CscXDocWords, w As Long  For r =1 To table.Rows.Count-1 'start on second line of table  Set row=table.Rows(r)  For c =0 To row.Cells.Count-1  Set cell=row.Cells(c)  Set cellAbove=table.Rows(r-1).Cells(c)  If cell.Text="" And cellAbove.Text<>"" Then  'Find the words in the cell above  Set words=pXDoc.GetWordsInRect(cellAbove.PageIndex, \_  cellAbove.Left,cellAbove.Top, \_  cellAbove.Width,cellAbove.Height)  For w =0 To words.Count-1  'This is the ONLY way to set the coordinates of a table cell  cell.AddWordData(words(w))  Next  End If  Next  Next  End Sub |

## A word about .Net and scripting

**Content to be supplied later**

## Useful COM Objects & Win32 Functions

|  |  |
| --- | --- |
| Object | Function |
| FileSystemObject | Deleting/Creating Folders. Winwrap is adequate for file input/output and Unicode |
| MicrosoftScriptingRuntime. | Dictionary. For creating lookup lists (also called hashes or associative arrays), counting words, removing duplicates, finding unique words, histograms. <http://msdn.microsoft.com/en-us/library/x4k5wbx4.aspx> |
| Microsoft VBScript Regular Expressions 5.5 | Useful for splitting strings. Be aware that regex fails on OCR errors. Good for using in validation rules. |
| Microsoft ActiveX Data Objects 6.1 Library | SQL queries. (The KTM SQL object only supports SELECT and not INSERT) http://msdn.microsoft.com/en-us/library/windows/desktop/ms681504(v=vs.85).aspx |
| Microsoft XML, v 6.0 | XML, XSL, Xpath, webservices, https, Kapow integration. |
| Private Declare Sub OutputDebugString Lib "kernel32" Alias "OutputDebugStringA" (ByVal msg As String) | Debugging with Sysinternals‘ DebugView |
| Private Declare Function QueryPerformanceCounter Lib "kernel32" (v As Long) As Boolean | Retrieves CPU time in the order or microseconds, but is not a true clock. Don‘t trust for timings. |
| Private Declare Function GetTickCount Lib "kernel32" () As Long | Retrieves the number of milliseconds (typical resolution of 10-16 ms) that have elapsed since the system was started, up to 49.7 days. |

# Barcodes

**Content to be supplied later – barcode types, dimensions, quiet zones, Datamatrix, redundancy, checksums.**

# Unicode

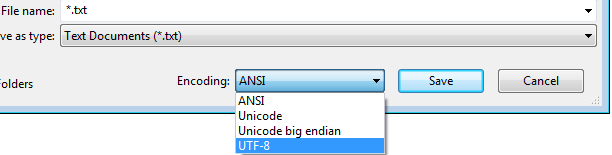
Kofax Transformation is fully Unicode compliant, supporting almost any character set, including right-to-left languages.

KTM has numerous XML configuration files which are all in Unicode.

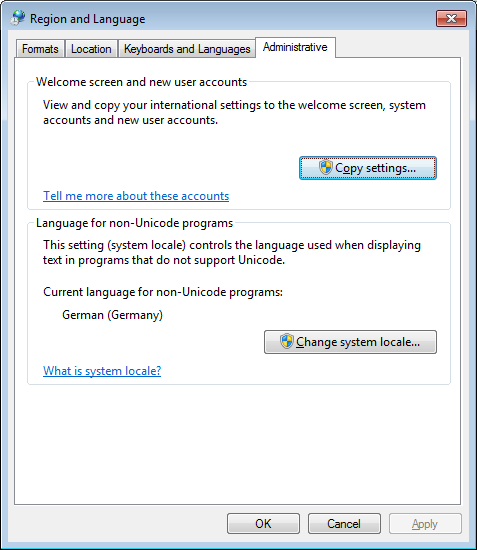
However you need to take care that your **Databases** and **Dictionaries** are also in Unicode. In today’s global world with large customer databases, you really should always use Unicode.

## Databases & Dictionaries

You should ALWAYS write your database and dictionary files using a Unicode encoding, either UTF-8 or UTF-16 (UTF-16 is also called Unicode). This will ensure that you have no character corruption or loss of data using characters outside of A-Z. Window’s **Notepad** can show and change the encoding in the File/SaveAs… Menu. **Unicode** is 16 bit. Use **UTF-8** for an 8 bit encoding and smaller file sizes.

In the image below from **Notepad.exe** the file is currently encoded in ANSI, and will be converted to UTF-8.  


When Kofax Transformation opens a database or dictionary file, it checks the first bytes of the file. If it finds a [**http://en.wikipedia.org/wiki/Byte\_order\_mark**](http://en.wikipedia.org/wiki/Byte_order_mark)(BOM)it will parse the file according to the BOM. The UTF-8 BOM is **0xEF,0xBB,0xBF** and is seen as **ï»¿** in non-Unicode Text Editors using the Character set "Latin alphabet no. 1," ISO-8859-1.

If there is no BOM in the text file then Kofax Transformation will use the “Current Language for non-Unicode programs” in **ControlPanel/RegionAndLanguage/Administration** settings. KTM Server will use the “System Locale”. If your vendor searches work in Project Builder but fail in KTM server, you have possibly not encoded your databases in Unicode.  


If you are exporting CSV databases for use as Fuzzy Databases, make sure that your system is inserting the BOM into the database files. If the system cannot do this, then you can use Search & Matching Server.

## Scripting Unicode

The KTM scripting environment fully supports Unicode, except for the Debugger Windows. You can add Unicode characters directly into scripts. filename="Παρθενώνας.txt"

KTM scripts will automatically detect the encoding of a text file when you read it. Use the following code to create a Unicode file.

Open Filename For Output As #1

Print #1, vbUTF8BOM;

## Corrupted File Names in Windows

Windows NTFS uses Unicode to store file names. However, if you unpack a zip file containing Russian or Greek filenames then the file names can be corrupted, and Project Builder will not be able to open the files. You have to change your regional settings before unzipping the files.

# Summary

I hope this book has helped you to build a great KTM solution and make your customer very happy and productive.

This book will continue to improve as you provide feedback about what was good, bad, helpful or missing to [field.enablement@kofax.com](mailto:david.wright@kofax.com)

# Appendices

## Document Collection With Kofax Express

This is your chance to obtain pristine images before people have stamped and marked them.

**Note**

**:** If you are doing an invoice project, then you should take barcode stickers with you, or ask the customer to provide them.

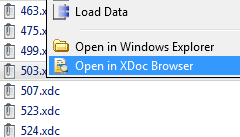
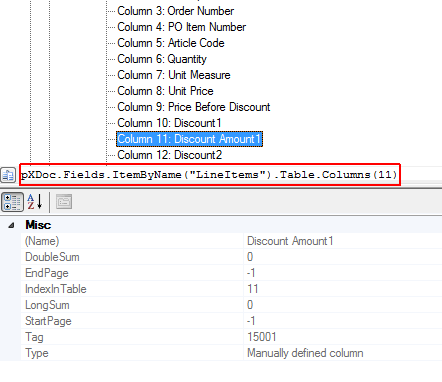
Kofax Express is excellent for this task because:

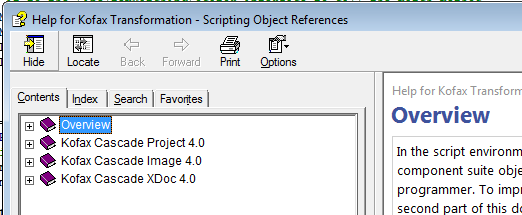
* You can take Kofax Express to your customer’s mailroom and scan documents before they stamp and mark them.
* It scans everything in color at the scanner’s rated speed.
* You can change batch and export settings on the fly.
* You can manually or auto- separate documents.
* You can adjust VRS settings on each document in the batch after you have finished scanning (Kofax Capture can manipulate only the last scanned image)
* VRS adjustments are immediately seen on the document.

Using Kofax Express

* Make sure you have 300 dpi (Kofax Express defaults to 200 dpi)
* Set up VRS with Deskew, Autocrop, Edge Cleanup, Hole Punch Removal, auto brightness, auto-clarity 3.  
  
* Set up your batch export to be Multi-TIFF.
  + then you don’t have page separation problems in KTM Project Builder.
* Set up your separation strategy, either fixed page, barcode, or manual.
* Scan documents at your customer.
* Do not export batches. Kofax Express can have many open batches.
* Go back home and check each document for VRS settings and document separation.
* Export your batches. They are now ready for use in KTM Project Builder.

## Useful KTM Objects and Properties

The two easiest way to learn about the KTM scripting objects is to open any XDocument with the XDoc Browser in the right click menu and expanding the XDocument  
 

You can also open **Help/Scripting Object Reference** in the Script Editor  


You can also look at the Product Document at **Help/Scripting Help** in the Script Editor to learn about Script Events.

|  |  |
| --- | --- |
| Expression | Comment |
| pXDoc.ExtractionClass | The className of a document |
| pxDoc.Pages(0).Words | All words on page 1 |
| pXRootFolder.DocInfos(3).XDocument | The 4th document in the folder |
| pXDoc.GetWordsInRect(pageIndex,Left,Top,Width,Height) | Find words in a rectangle on a page (also used by Lassoing |
| pXDoc.TextLines(23).Words(2) | The 3rd word on the 23rd line of the document |
| pLocator.Alternatives(0).Words.Append(word) | Append a word to the first alternative of a locator. This handles text and coordinates properly. |
| pXDoc.CDoc.Pages.Count | Number of graphic pages of a document |
| pXDoc.Pages.Count | Number of OCR text pages of a document |
| tmpXDoc.CopyPages(pXDoc, 0, 3) | Copy pages 1, 2 & 3 from one document to another |
| pXDoc.Reclassify(ClassName, Confidence) | Reclassify a document (Do this either in Document\_AfterClassify, ValidationForm\_AfterFieldChanged, ValidationForm\_ButtonClicked |
| pXDoc.Pages(2).TextLines.Count | The number of textlines on page 3 |
| pXDoc.Words.Count | The number of words in the document. |
| pLocator.ItemByName(“TL”).Alternatives(0).Table.Rows(2).Cells(3).Text | The 4th cell of row 3 of the Table Locator “TL” |
| DefaultAmountLocator, DefaultDateLocator, Project | Global objects available in any script |
| pXDoc.ParentFolder.DocInfos(pXDoc.IndexInFolder-1).XDocument | The previous document |
| pField.PageIndex | The page on which the field appears. If this is -1, then the Field has no coordinates. |
| Project.TableModels.ItemByName(TABLEMODELNAME).ModelColumns.Count | The number of columns in a TableModel |
| Project.GlobalColumns.ItemByID(tablemodel.ModelColumns(3).GlobalColumnID).DisplayNameLocalizations.Default | The localized Name of the 3rd column of a table model. |
| Project.Databases, Project.Dictionaries | The fuzzy and SQL Databases, and the Dictionaries |
| pXDoc.CDoc.Pages(2).Xres | The horizontal dots per inch DPI of page 3. |
| pXDoc.CDoc.Pages(1).GetImage() as CSCImage | Returns the pixel image of page 2 |
| pXdoc.CDoc.Pages(2).SourceFileName | The image filename of page 3 |
| pXDoc.FileName | The filename of the XDocument |
| pXDoc.ReplacePageSourceFile(filename,"TIFF",2,0) | Replace page 2 of a document with a TIFF image. |
| Project.ClassByName(pXDoc.ExtractionClass).ParentClass | The Parent Class of a document |

## Visual Basic 6 tips

Use With … end With to make your code readable.

Use For…Next instead of copy/pasting code.

Use the following to loop through “Total”, “SubTotal”, etc…

For each fieldname in split("Total SubTotal TaxRate1 TaxAmount1")

Do error checking at beginning of functions.

If condition then exit sub.

Use the setting '#Language "WWB-COM" to get keywords return, andalso, orelse, isnot.

All four keywords will reduce nesting and increase readability.

Have tight logic loops and functionalize what is in them for clarity and easy debugging.

The following example reads every word on a line, fuzzy matches it to a database and then builds a score. It is quite complex, but the complexity has been pushed off into another reusable function, and this loop is now very simple.

|  |
| --- |
| For w = 0 To textline.Words.Count-1  word=LCase(Trim(textline.Words(w).Text))  match=**DataBase\_SearchString**(databaseName,"headerword",word,conf)  score=score+conf\*Len(word)  Next |

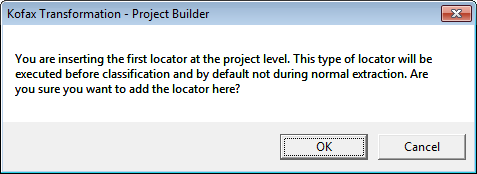
## Fuzzy Database Queries from Script

This is a useful and flexible script for retrieving one or many, fuzzy or precise results from a local or remote fuzzy database.

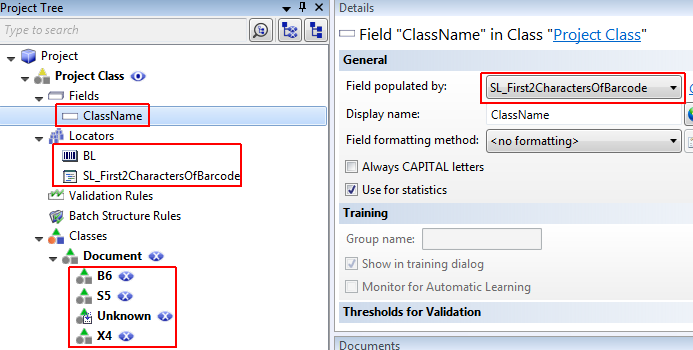
Combining an SQL Database with Kofax Search & Match Server and this script you can perform webqueries much faster and even perform fuzzy “webqueries.””.

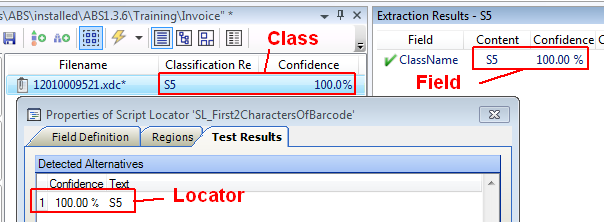
|  |
| --- |
| Public Function Database\_Search(dbname As String, column As String, searchString As String, numberHits As Integer, minimimConfidence As Double, Optional allColumns As Boolean=False) As CscXDocField  'Searches inside a fuzzy database for the searchstring and returns the results in the alternatives of a new CSCField Object.  'if column="" then all columns are returned as subfields, otherwise returns only the chosen column in the alternatives.  'Set minimimConfidence=1.0 for exact match search.  Dim DB As CscDatabase, Fields() As String,FieldIDs() As Long  Dim col As Integer,c As Integer,i As Integer  Dim hits As CscDatabaseResItems, alt As CscXDocFieldAlternative  Dim results As New CscXDocField 'You are allowed to create a standalone field  Dim value As String, substitute As String  Set DB=Project.Databases.ItemByName(dbname)  ' Replace all delimiters by blank  For i = 1 To Len(DB.AdditionalDelimiterChars)  searchString = Replace(searchString, Mid(DB.AdditionalDelimiterChars, i, 1), " ")  Next  ' Replace all ignore characters by blank  For i = 1 To Len(DB.RemoveChars)  searchString = Replace(searchString, Mid(DB.RemoveChars, i , 1), " ")  Next  ' Substitution pairs define which texts to be replaced by what.  For i = 0 To DB.SubstitutionPairCount - 1  DB.GetSubstitutionPair(i, value, substitute)  searchString = Replace(searchString, value, substitute)  Next  Fields = Split(searchString, " ")  ReDim FieldIDs(UBound(Fields))  'Find the column we are looking for  col=-1  For i =0 To DB.FieldCount-1  If DB.FieldName(i)=column Then col=i  Next  If col=-1 And column<>"" Then Err.Raise 34589,,"Column '" & column & "' does not exist in database '" & dbname & "'."  If col<>-1 Then 'Force query in this column  For c=0 To UBound(FieldIDs)  FieldIDs(c)=col  Next  End If  Set hits = DB.Search(Fields, FieldIDs, CscEvalMatchQuery, numberHits)  For i = 0 To hits.Count-1  If hits(i).Score>= minimimConfidence Then  Set alt= results.Alternatives.Create()  alt.Confidence=hits(i).Score  If allColumns Then 'the column is "", so we return all fields  For c = 0 To DB.FieldCount-1  alt.SubFields.Create(DB.FieldName(c))  alt.SubFields(c).Index=c  alt.SubFields(c).Text=DB.GetRecordData(hits(i).RecID)(c)  alt.SubFields(c).Confidence=hits(i).Score  Next  alt.Text=""  Else  alt.Text=DB.GetRecordData(hits(i).RecID)(col)  End If  End If  Next  Return results  End Function |

## How to Classify by Barcode in Script

You can use Locators attached to the Project Class level are classify documents. If these locators require OCR, then OCR will be performed at this stage.  


This example below uses the first two characters of the barcode on the document to classify the document as either “B6” or “S5” or “X4”. If no barcode is found, then the document is classified as “Unknown”.



Always use a script locator and put your script logic into it. This makes it easy for you to test, and it makes your project easy to understand as in the screenshot above. Make sure that ClassName is a project level field that will contain the confidence and the class name that you want.desire.  
 

Put this script into the project level script. The event Document\_AfterClassifyXDoc can reclassify the xdocument.

**Note:** These locator and field values are not available during extraction. To pass values from classification scripts to extraction scripts, then you will need to use CSCXDocument.XValues to store them. Remember: that Extraction runs after Document Review in KTM.

|  |
| --- |
| Private Sub SL\_First2CharactersOfBarcode\_LocateAlternatives(ByVal pXDoc As CASCADELib.CscXDocument, ByVal pLocator As CASCADELib.CscXDocField)  Dim Barcodes As CscXDocFieldAlternatives  Set Barcodes=pXDoc.Locators.ItemByName("BL").Alternatives  'If Barcodes.Count=0 Then Exit Sub ' No barcode was found, so we cannot classify  'If Len(Barcodes(0).Text)<2 Then Exit Sub 'This is not a valid barcode for classification  'Create an output alternative for this script locator  With pLocator.Alternatives.Create  .Text=Left(Barcodes(0).Text,2)  .Confidence=Barcodes(0).Confidence 'Copy the confidence from the barcode locator (it will always be 100%)  End With  End Sub  Private Sub Document\_AfterClassifyXDoc(ByVal pXDoc As CASCADELib.CscXDocument)  Dim c As Long, ClassName As CscXDocField  Set ClassName =pXDoc.Fields.ItemByName("ClassName")  'See if the field ClassName contains the name of a class. if so reclassify the document to it  For c =1 To Project.ClassCount ' ClassID is NOT zero-based.  If Project.ClassByID(c).Name=ClassName.Text Then  pXDoc.Reclassify(ClassName.Text,ClassName.Confidence)  Exit Sub  End If  Next  'leave the classification result alone  End Sub |

## How to Classify by Phrase in a Region on a Document

Instructional Classification looks everywhere on the document. Maybe you have a phrase at the bottom of the first page which clearly identifies the class.

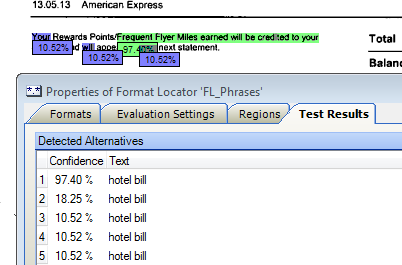
This method uses an auto-replace dictionary in a format locator to fuzzy match **long phrases** in a region. A great advantage of using such a dictionary over instructional classification is that the dictionary is outside of the project and can be managed by the customer directly.

Create a dictionary of all of these phrases with **Auto Replace Values** of the class name.

|  |
| --- |
| We appreciate your prompt payment and value you as our customer ; telephone bill  Frequent Flyer Miles will be credited to your;hotel bill |

Here you can see that the Format Locator has fuzzy matched the phrase, as well as other words with lower confidence, and returned the ClassName of “hotel bill” each time.

Dictionaries are case-insensitive and you need to ignore alternatives with scores below about 80%



Follow the same method as in the previous example with barcodes.

## SQL Queries from Script

KTM natively uses fuzzy databases. These are generally faster and simpler to use and configure than SQL.

In almost all cases, a fuzzy database is better than an SQL database.

* Is it possible that your query contains OCR or other errors? Use Fuzzy Database.
* Do you want to be able to search in any column in the database? Use Fuzzy Database.
* Do you want to keep SQL user password and connection string safe from client side scripts? Use Fuzzy Databases.
* Do you want to do an exact query and not a fuzzy query? Use Fuzzy Query with confidence set to 100%.
* Do you need to do an INSERT? Use SQL. But first ask yourself is this INSERT must happen during KTM Server or KTM Validation, and whether it would not be better to delay it until after KTM, in a BPM engine.
* Do you need to do interactive purchase order queries for the Line Item Matching Locator? Use SQL
* Do you need to look up long numeric values like customer IDS in databases with millions of records? Use SQL.. A fuzzy database query may be slower or even possibly miss records if there are many hundreds of entries in the database which differ by only 1 or 2 characters.

### 64 bit ODBC Connection

**Use %systemroot%\SysWOW64\odbcad32.exe to create ODBC connections on 64 bit Windows 7.**

### Using CscSQLDataTable for SQL “SELECT”

KTM has scripting objects CscSQLDataTable, CscSQLQuery and CscSQLRecordSet that you can use for performing “SELECT” queries. This object has the advantage that you do not need to create, hold and destroy the connection to the database.

“INSERT” is not supported. You cannot connected to a database on Search & Matching Server.

|  |
| --- |
| 'KTM supports SQL SELECT queries natively  Dim SQLDataTable As CscSQLDataTable, SQLQuery As CscSQLQuery  Dim r As Long, Recordset As CscSQLRecordset  Set SQLDataTable = Project.Databases.ItemByName("VendorsSQL").SQLTable  Set SQLQuery = SQLDataTable.CreateQuery()  SQLQuery.AddSelectField(SQLDataTable.FieldByName("VendorID"))  SQLQuery.AddSelectField(SQLDataTable.FieldByName("VendorName"))  SQLQuery.AddWhereField(SQLDataTable.FieldByName("VendorID"), CscEqual, "123")  Set Recordset = SQLQuery.ExecuteQuery()  For r = 0 To Recordset.RecordCount-1  With Recordset.RowByIndex(r).Cells  'Do something with results  End With  Next |

### Using ActiveX Data Objects for SQL “INSERT” and others

Add a reference to “Microsoft ActiveX Data Objects 6.1 Library” to your script.

You will need to manage opening and closing connections yourself. You can use the project level script event **Application\_InitializeScript** and **Application\_DeinitializeScript** to open and close these once per session. Be aware that in KTM Server all Document Worker Threads will be using this same object in parallel.

See MSDN for code examples. <http://msdn.microsoft.com/en-us/library/windows/desktop/ms675947(v=vs.85).aspx>

## Calling a Fuzzy Database in a Field Formatter for Spell Checking

This script calls a fuzzy database and checks if there are 0, 1 or more results and decides if the results are “good enough” for a match

It uses the **Database\_Search** function from Visual Basic 6 tips

Use With … end With to make your code readable.

Use For…Next instead of copy/pasting code.

Use the following to loop through “Total”, “SubTotal”, etc…

For each fieldname in split("Total SubTotal TaxRate1 TaxAmount1")

Do error checking at beginning of functions.

If condition then exit sub.

Use the setting '#Language "WWB-COM" to get keywords return, andalso, orelse, isnot.

All four keywords will reduce nesting and increase readability.

Have tight logic loops and functionalize what is in them for clarity and easy debugging.

The following example reads every word on a line, fuzzy matches it to a database and then builds a score. It is quite complex, but the complexity has been pushed off into another reusable function, and this loop is now very simple.

|  |
| --- |
| For w = 0 To textline.Words.Count-1  word=LCase(Trim(textline.Words(w).Text))  match=**DataBase\_SearchString**(databaseName,"headerword",word,conf)  score=score+conf\*Len(word)  Next |

Fuzzy Database Queries from Script.

|  |
| --- |
| Private Sub USState\_FormatField(ByVal FieldText As String, FormattedText As String, ErrDescription As String, ValidFormat As Boolean)  Dim results As CscXDocField  If Len(FieldText) = 0 Then  ValidFormat = False  ErrDescription = "Field must not be empty"  Exit Sub  End If  'Look for up to 5 fuzzy database matches with scores better than 80%  results =Database\_Search("USStates",FieldText,"",5,0.8)  Select Case results.Alternatives.Count  Case 0  ErrDescription= FieldText & " is an unknown US State"  ValidFormat=False  Case 1  FormattedText=results.Alternatives(0).Text  ValidFormat=True  Case Is >1  If results.Alternatives(0).Confidence-results.Alternatives(1).Confidence>0.3 Then  'the first Result is much better than the second result  FormattedText=results.Alternatives(0).Text  ValidFormat=True  Else  'Show two options in the error message  ErrDescription = results.Alternatives(0).Text & ";" & results.Alternatives(1)  ValidFormat=False  End If  End Select  End Sub |

## Calling a Fuzzy Database in a Validation Rule

This script is very much like the formatting script in the previous appendix, but it doesn’t change the value and only accepts perfect matches.

|  |
| --- |
| Private Sub USState\_Validate(ByVal pValItem As CASCADELib.ICscXDocValidationItem, ByRef ErrDescription As String, ByRef ValidField As Boolean)  Dim results As CscXDocField  If Len(pValItem.Text) =0 Then  ValidField = False  ErrDescription = "Field must not be empty"  Exit Sub  End If  'Look for up to 1 perfect database matches with scores =100%  results=Database\_Search("USStates",pValItem.Text,"",1,0.8)  If results.Alternatives.Count=0 Then  ValidField = False  ErrDescription = "Field must not be empty"  Exit Sub  ElseIf results.Alternatives(0).Confidence=1.0 Then  'Perfect Match  ValidField=True  Else  'Suggest the best match  ErrDescription="Did you mean " & results.Alternatives(0).Text & "?"  ValidField=False  End If  End Sub |

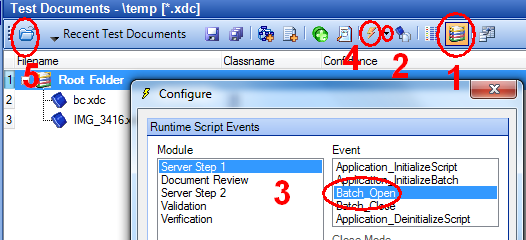
## Create Golden Files from a Productive System

Add the following script to the project level script and republish the batch in Kofax Capture. It will save all XDocs both after KTMServer and after KTM Validation. You can then benchmark the project against the validation files as well as benchmarking the files from after KTM Server, which will show you if Project Builder has different results as the Productive System.

|  |
| --- |
| Private Sub Batch\_Close(ByVal pXRootFolder As CASCADELib.CscXFolder, ByVal CloseMode As CASCADELib.CscBatchCloseMode)  'Create golden files in production system  'make sure KTM Server and KTM Validation have write access to the following folder.  XFolder\_Backup(pXRootFolder,"\\Server\GoldenFiles\")  End Sub  Public Sub XFolder\_Backup(ByVal XFolder As CASCADELib.CscXFolder, BackupFolder As String)  'Add reference to "Microsoft Scripting Runtime" for FileSystemObject  Dim fso As New FileSystemObject, source As String, BatchGUID As String  'Use this for creating golden files form a productive KC-KTM system  'Check which Module we are in.  Select Case Project.ScriptExecutionMode  Case CscScriptExecutionMode.CscScriptModeValidation  'Check that we are in Validation Step 2  'If Project.ScriptExecutionInstance<>2 Then Exit Sub  BackupFolder=BackupFolder & "AfterValidation\"  Case CscScriptExecutionMode.CscScriptModeServer  'Check that we are in KTM Server 2  'If Project.ScriptExecutionInstance<>2 Then Exit Sub  BackupFolder=BackupFolder & "AfterKTMServer\"  Case Else  Exit Sub 'DocReview, Character Correction, Knowledge Base Modules  End Select  BatchGUID=XFolder.XValues.ItemByName("AC\_BATCH\_GUID").Value  BatchGUID=Replace(BatchGUID, "{","")  BatchGUID=Replace(BatchGUID, "}","")  source=fso.GetParentFolderName(XFolder.FileName)  If Not fso.FolderExists(BackupFolder) Then fso.CreateFolder(BackupFolder)  fso.CopyFolder(source,BackupFolder & BatchGUID,True) 'Copy the entire KC batch folder  Set fso = Nothing  End Sub |

## Create Golden Files from CSV Data

* Create a text file where the first row consists of fileName and all field headers separated by semicolons. Each other line contains the full path name to the Xdoc file and then all field values separated by semicolons.
* Load the Xdocs into project builder.
* Put the code below into the Project level script.

1. Switch to Batch View  
   
2. Open Test Runtime Script Events/Configure**….** (small black triangle)
3. Select event Batch\_Open
4. Press the lightning icon (Ctrl-F11)
5. Reload the Xdocs.

Code for the Project level script:

|  |
| --- |
| Private Sub Batch\_Open(ByVal pXRootFolder As CASCADELib.CscXFolder)  Folder\_ImportCSV(pXRootFolder,"c:\temp\goldenvalues.csv")  End Sub  Const CSVSEP=";"  Private Sub Folder\_ImportCSV(ByVal pXRootFolder As CASCADELib.CscXFolder, csvFileName As String)  'Include Reference to “Microsoft Scripting Runtime”  'This imports from a CSV file all the values into the XDocs  'The first column MUST contain the fully qualified XDoc FileName.  'The second column MUST contain the class name.  'The first row MUST contain “filename;classname;fieldname1;fieldname2;…”  'Non-existing field values are ignored  Dim x As Integer, f As Integer  Dim xdoc As CscXDocument  Dim txt, key, values() As String  Dim dict As New Dictionary  Dim fieldIds As New Dictionary  Open csvFileName For Input As 1  Line Input #1, txt 'First line of csv file  For Each key In Split(txt, CSVSEP) 'make an index of field names to fieldID  fieldIds.Add(key, f)  f=f+1  Next  While Not EOF(1)  Line Input #1, txt 'Load the rest of the csv file into a dictionary with filename as key.  values=Split(txt,CSVSEP,2)  dict.Add(values(0),values(1)) 'this makes the XDoc filename the key and  ' “classname;fieldname1;fieldname2;…” the value  Wend  For x = 0 To pXRootFolder.DocInfos.Count-1  Set xdoc=pXRootFolder.DocInfos(x).XDocument  If dict.Exists(xdoc.FileName) Then  values=Split(dict.FileName,CSVSEP)  xDoc.Reclassify(values(0))  For f =0 To xdoc.Fields.Count-1  If fieldIds.Exists(xdoc.Fields(f).Name) Then xdoc.Fields(f).Text=values(fieldIds(xdoc.Fields(f).Name)-1)  Next  End If  xdoc.Save  Next  End Sub |

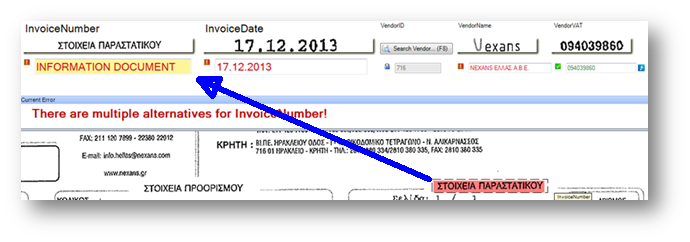
## Integrating Webservices & Kapow with KTM

[Kapow Software’s](http://kapowsoftware.com/) synthetic APIs brings immense value to a KTM project.

Some points of integration

* Validation Rules checking with the internet.
* Lassoing Text on a Document.
* Creating Fuzzy Database Files from Websites.

The following Code calls Google Translate via Kapow to translate text on a document using the ValidationForm\_AfterViewerLassoDrawnevent.

In the example below the user has lassoed “ΣΤΟΙΧΕΙΑ ΠΑΡΑΣΤΑΤΙΚΟΥ” on the image and the translation “INFORMATION DOCUMENT” is inserted into the field. 

|  |
| --- |
| Private Sub ValidationForm\_AfterViewerLassoDrawn(ByVal pXDoc As CASCADELib.CscXDocument, ByVal PageIndex As Long, ByVal pField As CASCADELib.CscXDocField, ByVal TopPos As Long, ByVal LeftPos As Long, ByVal Width As Long, ByVal Height As Long, ByRef bCancel As Boolean)  Dim words As CscXDocWords, translation As String, ms As Long  Set words=pXDoc.GetWordsInRect(PageIndex,LeftPos,TopPos,Width,Height)  If words.Count=0 Then Exit Sub 'The user lassoed no words  translation=Kapow\_GoogleTranslate("http://localhost:50079/rest/ru2n/Default project/GoogleTranslate\_rpb",words.Text,"Greek","English", ms)  pField.Text=translation  pField.PageIndex=PageIndex  pField.Left=LeftPos  pField.Top=TopPos  pField.Width=Width  pField.Height=Height  bCancel=True 'Tell KTM to ignore the lasso event because Kapow handled it!  End Sub  Public Function Kapow\_GoogleTranslate(url As String, sourceText As String,sourceLang As String, destLang As String, ByRef milliseconds As Long) As String  'Add reference to "Microsoft XML 6.0"  Dim xml As New MSXML2.DOMDocument60, responseXML As MSXML2.DOMDocument60  Set xml=Kapow\_CreateRESTRequest()  Kapow\_AddParameter(xml,"Text","translate.SourceLanguage",sourceLang)  Kapow\_AddParameter(xml,"Text","translate.DestinationLanguage",destLang)  Kapow\_AddParameter(xml,"Text","translate.SourceText",sourceText)  Kapow\_AddParameter(xml,"Text","translate.DestinationText","")  Set responseXML=HTTP\_POST(url, xml)  If responseXML.selectNodes("//rest-result").Length=0 Then Return "Error: No Internet Connection!"  milliseconds=CInt(responseXML.selectSingleNode("//rest-result/@executionTime").Text)  Return responseXML.selectSingleNode("//attribute[@name='DestinationText']").Text  End Function  Public Function Kapow\_CreateRESTRequest() As MSXML2.DOMDocument60  Dim xml As New MSXML2.DOMDocument60  xml.appendChild(xml.createProcessingInstruction("xml", "version=""1.0"" encoding='UTF-8' standalone='yes'"))  xml.appendChild(xml.createElement("rest-request" )).appendChild(xml.createElement("parameters"))  Return xml  End Function  Public Sub Kapow\_AddParameter(xml As MSXML2.DOMDocument60, ParameterType As String, Parameter As String, value As String)  Dim vars() As String, variable As MSXML2.IXMLDOMNode  vars=Split(Parameter,".")  Set variable=xml.selectSingleNode("//variable[@variableName='"& vars(0) & "']")  If variable Is Nothing Then  Set variable= xml.selectSingleNode("//parameters").appendChild(xml.createElement("variable"))  variable.attributes.setNamedItem(xml.createAttribute("variableName")).Text=vars(0)  End If  With variable.appendChild(xml.createElement("attribute"))  .attributes.setNamedItem(xml.createAttribute("type")).Text=ParameterType  .attributes.setNamedItem(xml.createAttribute("name")).Text=vars(1)  .Text=value  End With  End Sub  Public Function HTTP\_POST(url As String, xml As MSXML2.DOMDocument) As MSXML2.DOMDocument60  Dim XMLHTTP As New MSXML2.XMLHTTP60  XMLHTTP.Open("POST", url, False) 'You can also add username&password here for http basic authentication  XMLHTTP.setRequestHeader("Content-Type", "text/xml")  XMLHTTP.send(xml)  If XMLHTTP.status<>200 Then 'Some error happened  With XMLHTTP.responseXML.appendChild(xml.createElement("status"))  .Text=XMLHTTP.statusText  .attributes.setNamedItem(xml.createAttribute("value")).Text=CStr(XMLHTTP.status)  End With  End If  Return XMLHTTP.responseXML  End Function |

## Converting Fields to Dates and Amounts

You should avoid using the following functions in your scripting CDbl, CLng, CDate, CCurr as they can have very unpredictable results.

You have to assume that the input to these functions has come from OCR and hence contains errors. You also have to assume that you don’t know what the regional settings are on the client machine – you may not really know if the user has dates MM\_DD or DD\_MM. You don’t always know what the decimal and thousands separator is.

It is recommended to use these functions for converting Strings into Dates and Amounts. They use the standard formatters that you configure in your project. This keeps all of your number and date parsing consistent, and presents consistent error messages to the user.

The functions below always return a value. If the date is unformatted then you have an undefined value. If you want to check that the value formatted correctly check the Boolean values f.DoubleFormatted or f.Dateformatted.

|  |
| --- |
| Private Function String\_ParseDouble(a As String,Optional amountFormatter As String) As Double  Dim f As New CscXDocField  If amountFormatter="" Then amountFormatter=Project.DefaultAmountFormatter  f.Text=a  Project.FieldFormatters.ItemByName(amountFormatter).FormatField(f)  Return f.DoubleValue  End Function  Private Function String\_ParseDate(a As String,Optional dateFormatter As String) As Date  Dim f As CscXDocField  If dateFormatter="" Then dateFormatter=Project.DefaultDateFormatter  f.Text=a  Project.FieldFormatters.ItemByName(dateFormatter).FormatField(f)  Return f.DateValue  End Function |

# Index

"In-Place" Locator Scripting 88

.Net Scripting 89

A2iA 56

ADO

ActiveX Data Objects 104

Advanced Evaluator 57, 60

Advanced Zone Locator 35, 56

Amount Group Locator 56

Amount-based Algorithm 69

Anchors

Changing Training Anchors 58

**Application\_InitializeScript** 104

Auditing a Project 82

**Autocomplete** 38

Automatic Document Separation 53

Automatic Table Locators 67

Barcode Locator 56

Barcodes 90

Batch Restructuring 81

Batch Splitting 82

Benchmarking 23

Productive System 109

Table Data 25

Character Level Confidence 21

Classification

by Barcode 101

by region of a page 51, 103

Classification Trees 44

Clustering 47

Color

Change Color of a Label 39

COM Objects 89

Conflicts

Resolving 60

Customer Analysis 12

Database Evaluator 55

Spell checking 30

Uses 63

Database Formatter 30

Database Locator 55

Databases 62

Using fuzzy databases in scripts 50, 99, 106

Datamatrix 53

Dates

US & European 79

Debugging 83

DebugView 90

**Delivery Number** 78

Deterministic Software

Definition 11

Dictionaries 78

Dictionary 35

**Discrepancies** 41

Divide & Conquer 83

Divide and Conquer 52

Document Separation 52

**Document\_AfterExtract** 84

Embedded Columns 71

Error handling 83

Excel

Using Microsoft Excel to collect Data 17

False Positives 9, 22

Field Analysis Workshop 18

Field Viewers 39

Find and Filter 52

Fonts 39

Format Locator 35, 57

With Fuzzy Dictionaries 55

Formatter

using Fuzzy Databases 65, 106

Function Keys 38

Fuzzy Database vs ODBC 65

Goals, Needs, Pains 13

Golden Files 32

Creating from CSV File 110

From a Productive System 109

used for building a Benchmark 23

Google Translate 112

Header-based Algorithm 70

Hotkeys 37

How Many Documents? 26

Inheritance 44

Instructional Classification 48

Interleaved Columns 71

Invoice Evaluator 57, 60

Invoice Group Locator 55

Invoice Header Locator 57

Kapow

Integrating with KTM 90

Scripting Example 112

Key Value Pairs 18

Kofax Export Connector 32

Kofax Express 93

Kofax Web Capture 29

Lasso 112

Layout Classification 48

Learning 33

Learning Software

Definition 11

Line Item Matching Locator 55, 72

Line-based Algorithm 70

Localization 74

Locator

"In Place" Scripting 88

Locators 33, 54

Manual Table Locator 70

Margin of Error 26

Metrics 21

Accuracy Metrics 21

OCR Reread 29

OCR Voting 57

ODBC

Creating Entries on 64 bit machines 104

Online Training 59

Order Group Locator 56

Passive Information 41

Performance Metrics 23

Position-based Algorithm 70

Priorities 23, 34

Reject Documents and Pages 81

Relational Evaluator 57

Remote Fuzzy Databases 66

Representative Documents 16, 26

Scoring 59

Script Classification 49

Script Events 96

Script Locator

Multiple Subfields 86

Single Subfield 86

Scripting 83

Tables 88

Search And Matching Server 65

Searching

Validation Form 64

Separator Sheets 53

Skip Modules 81

Spell Checking 63, 106

Spellchecking 30

SQL 66

Comparison with Fuzzy 104

Queries from Script 104

Scripting 104

Using CscSQLDataTable 104

Square Root of N 26

Standard Evaluator 57

Statement of Work 21

Steps in Building a Project 34

StrongPhrases 78

Structural Classes 45

Table Header Pack 69

Table Interpolation 74

Table Locator 56

Table Model 68

Tables 67

Scripting 88

Text Classification 48

Text Content Locator 56

the Line Item Matching Locator

Field Dependencies 74

Trainable Document Separation 48, 51, 53

Trainable Group Locator 56

True Positives 9, 22

Unicode 90

User Experience 36

Validation Rule

Fuzzy Databases 108

Vendor Locator 29, 57, 62

Vision 13

Visual Basic

Tips 49, 98, 106

VRS 29, 93

Walkthrough 15

WeakPhrases 78

Webservice 64

Webservices

Disadvantages 66

Scripting Example 112

Win32 Functions 89

Windows Explorer 46

Workflow Agents 81

1. The foundation and the approach to customers of this section were inspired by a “Competitive Communications” Workshop conducted by Ralf Greis from St. Gallen Institute. <http://www.stgallen-institute.ch/partner/ralf-greis/> [↑](#footnote-ref-2)
2. See Appendix 1 for how to use Kofax Express to produce high- quality representative documents. [↑](#footnote-ref-3)