Automatic Record Matching in the Payment Reconciliation Journal

Design Insight
White Paper

Ciprian Iordache
Software Development
Engineer II

Nikola Kukrika
Software Development
Engineer II

September 2014

This white paper explains how automatic record matching is performed in the Payment Reconciliation Journal window where bank transactions are matched with open customer and vendor ledger entries that the related payments can be applied to.

Note that this record matching is different from automatic record matching in the Bank Acc. Reconciliation window where we match bank transactions with open bank ledger entries.

Contents

Problem Statement

The first requirement we had to meet was to match open customer/vendor ledger entries with transactions in a bank statement file. The second requirement was to provide a mapping of specific payment text to specific accounts for direct posting of payments without applying, such as recurring expenses without invoices.

The goal was to create a generic algorithm that provides matches with high confidence in most cases and clearly marks the cases where the match is made with lower confidence.

Since information in the bank statement file is different for each bank and the process of closing open entries is different for each company, it was important that the algorithm can be extended to fit different companies with minimal modification. One of the design considerations was to support multitenancy scenarios where different companies run on the same code base.

To trust the system, the user must understand how the system works. Therefore, a number of indicators in the UI explain how the matching is done.

The algorithm is implemented to apply the best match first. We believe this is a better solution than to try to maximize the match score for all entries as that would result in more wrong applications and weaker performance.

General Principle

Let A and B be the datasets that we must match. Let A contain n records and B contain m records. Let p be the number of attributes that are compared for the two datasets. The attributes are shared by the two datasets, either as identical or corresponding attributes.

The algorithm works as follows:

1. Filter datasets A and B to contain only the minimal set of records that could represent matches. (Assume that n records in A and m records in B remain after filtering.)
2. Loop through every pair of records inside the Cartesian product, A x B. For each couple, (A(i),B(j)), where I goes from 1 to n and j goes from 1 to m, do the following:
   a) Decide if (A(i),B(j)) should be excluded from the list of potential matches based on specific criteria.
   b) If the couple is not to be excluded, iterate and assess each of the common attributes, to see if they match fully/partially or do not match.
   c) Identify the match score (from a table of configured payment application rules, Bank Pmt. Appl. Rule table) based on the assessed p attributes.
   d) Save the couple as a match candidate, including the score in a temporary table.
3. Let C be the set of match candidates that resulted from step 2. Every record in C represents a pair of elements A(i), B(j) and their score.
4. Sort C descending on the score value. For each C(k), apply A(i) to B(j) as long as neither A(i) nor B(j) have previously been applied to any other element in the candidates set.

Note: This is a heuristic algorithm, which produces a solution quickly enough and well enough to solve the problem at hand.

The payment application rules table is configurable, so the rules can be adapted to different companies without code modifications and to support multitenancy scenarios. The algorithm and the payment application rules table also provide extension points, so additional matching attributes can be added with minor modification to the code.

Implementation

Before starting the algorithm, all bank statement lines are transferred to the Bank Acc. Reconciliation Lines table by using the Data Exchange Framework.

Note: The lines in the Bank Acc. Reconciliation Lines table are hereafter referred to as "bank transactions".

The algorithm performs the following overall phases:

1. Initialization
2. Matching
3. Applying

The algorithm can run in the following modes:

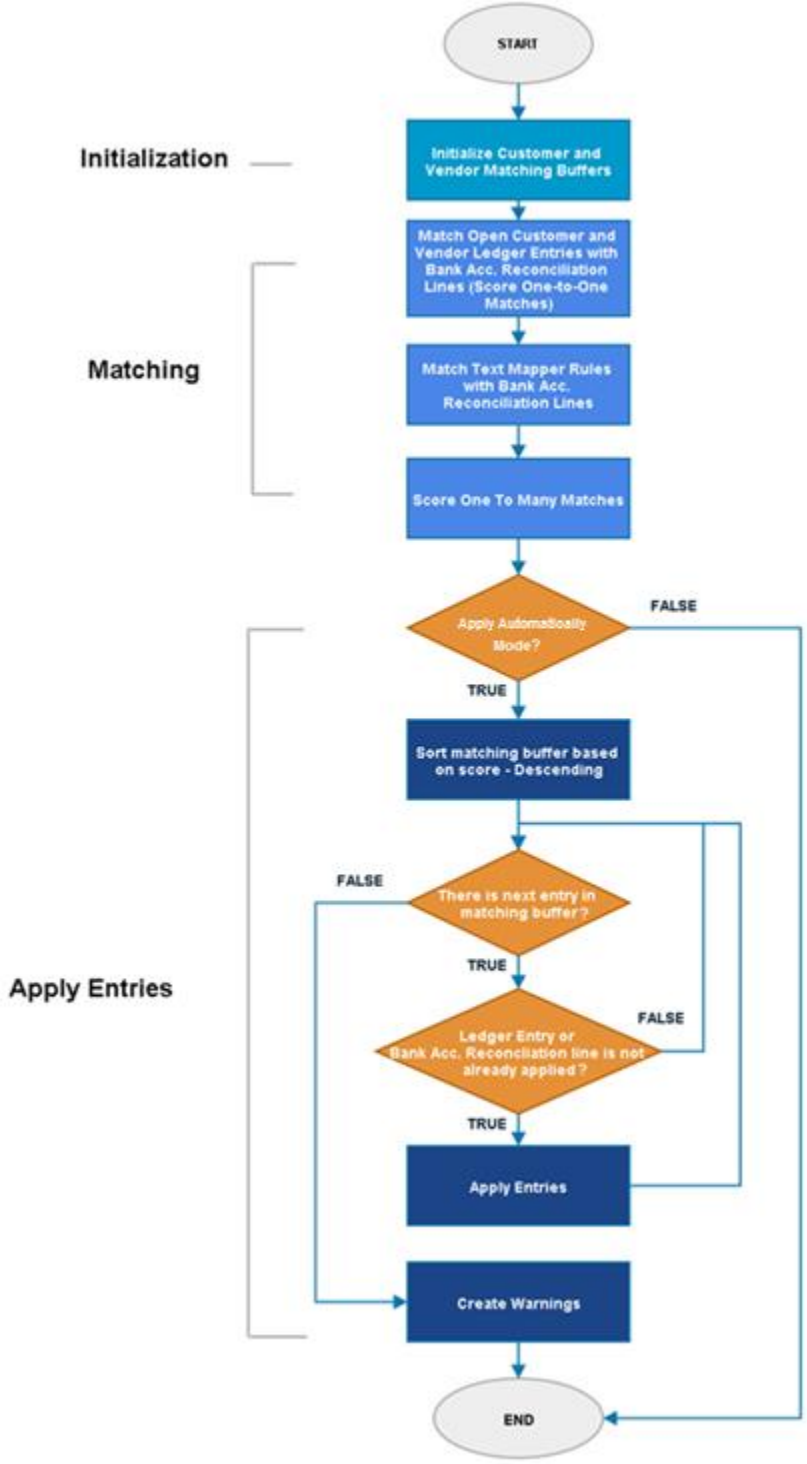
- *Apply Automatically* (ApplyEntries is set to TRUE) – The algorithm matches and applies entries. If this function is run twice, it will first remove all existing applications and then make new ones.
- *Propose Entries* (ApplyEntries is set to FALSE) – The algorithm only matches entries and sets a match confidence. This mode is used for review and manual application in the Payment Application window.

The implementation consists of the following general steps:

1. Initialize all needed structures for matching
2. Compare each bank transaction with open customer/vendor ledger entries.
3. Assign a score based on how many attributes match according to the payment application rules.
4. Save each couple as a match candidate in a temporary table.
5. If the mode is *Apply Automatically*, then sort the match candidates by score descending.
6. Go through all match candidates and apply them if the bank transaction or open entry is not already applied.

Note: If the mode is not *Apply Automatically*, then no sorting of the result is done. Sorting is then done outside of the algorithm where the results are used, such as in the Payment Application window.

The following diagram shows the overall phases.



Initialization Phase

In the initialization phase, we clear all temporary records and load customer and vendor ledger entries into temporary run-time versions of the Ledger Entry Matching Buffer table that are used in the matching phase. This design improves performance and supports matching with payment discounts. In addition, the temporary tables serve as good extension points for adding new matching criteria.

When loading entries into the temporary tables, the following steps are performed:

1. Assign the correct remaining amount:
    a. For accounts in a foreign currency, only invoices in that currency are loaded.
    The Remaining Amount field of the ledger entry is used for this.
    b. For accounts in the local currency, all invoices are loaded.
    This can be disabled by selecting the Appln. Between Currencies check box in the Sales & Receivables Setup window. All amounts, including payment discounts, must be converted to LCY.
2. Set the Payment Discount Date field and calculate the Remaining Amount Incl. Discount field.
    This is needed to match open entries that have payment discounts. The payment discount date is the latest of the dates in the Pmt. Disc. Tolerance Date and Pmt. Discount Date fields.

The payment application rules and the match score are defined in the Bank Pmt. Appl. Rule table (1252). This table is implemented as a temporary table for the following performance reasons:

- During matching, a lot of SQL queries are executed due to the complexity of the algorithm and the volume $O(n^2)$ (the number of open entries x number of bank transactions). Many of these queries are not cached. Because the data is loaded in the temporary table, it will be in memory and therefore the algorithm will not invoke SQL.
- Querying a temporary table has higher performance than querying SQL if the record set is smaller than 50.000 records.
- Only the needed data is loaded, which means that the data in memory is small.
- The calculation of the remaining amount from customer and vendor ledger entries is expensive.

Matching Phase

In the matching phase, the algorithm populates the Bank Statement Matching Buffer table with match candidates. Each match candidate will be assigned a score. For each bank transaction, the algorithm first finds and inserts matches with open entries and then finds and inserts matches according to text-to-account mappings. See the "Text-to-Account Mapping" section.

One-to-many matches are identified based on document number. To be able to score one-to-many matches, the algorithm must identify all document number matches for a given bank transaction and then score them using the same method as for one-to-one matches. For this reason, the scoring of one-to-many matches is done at the end. See the "One-to-Many Matching" section.

For each bank transaction, the algorithm iterates over all entries in the TempCustomerLedgerEntryMatchingBuffer and TempVendorLedgerEntryMatchingBuffer run-time tables with following steps:

Check if the Bank Transaction can Match an Open Entry

To increase the accuracy of automatic application, the algorithm excludes all entries that have a different sign than the sign on the bank transaction and a date that is before the posting date of the entry.

For review and manual application, no check is made and all entries are included because the user must be able to see all existing entries. A warning is displayed if the user tries to apply to an entry with a different sign or a late posting date.

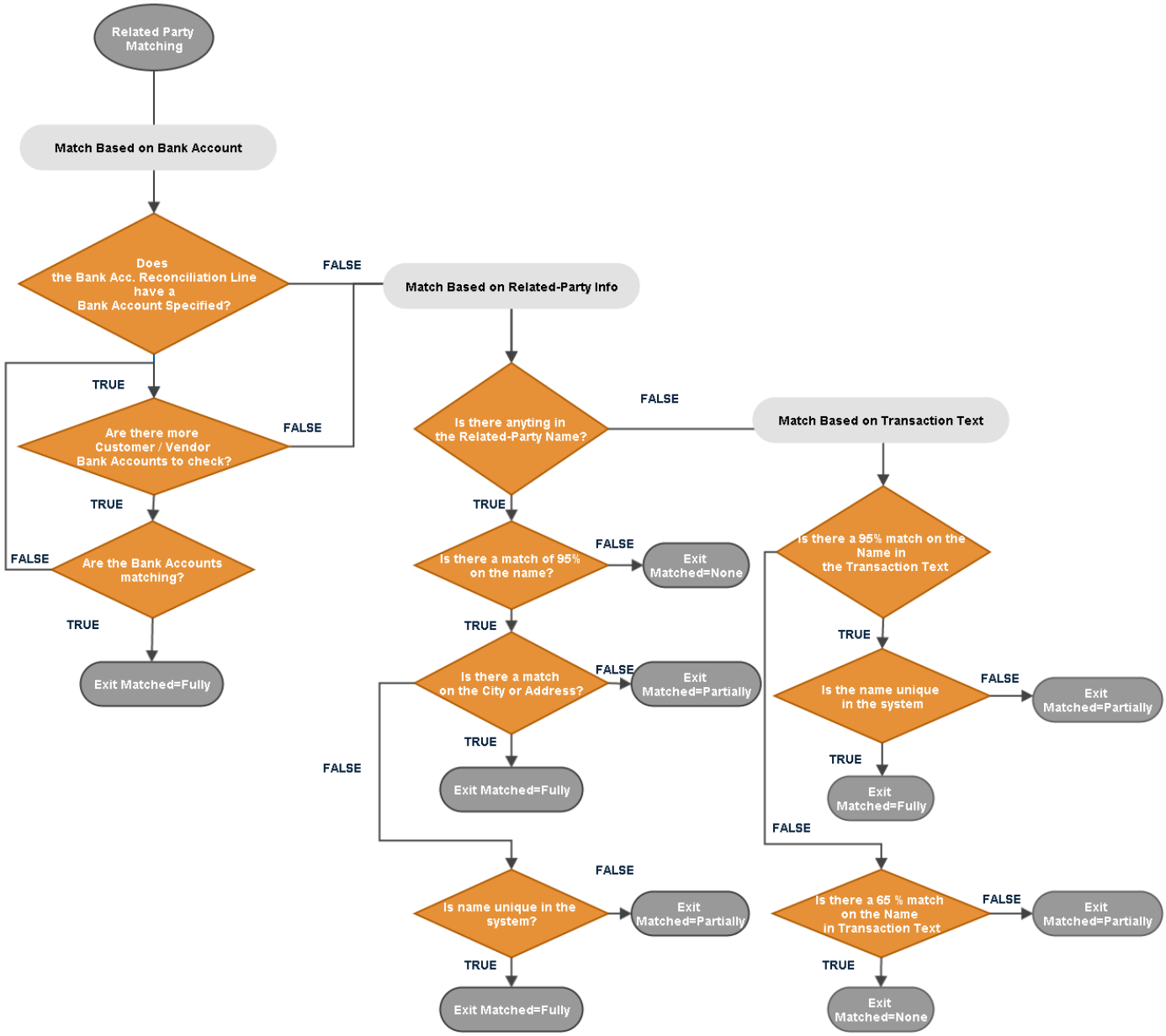
Match on the Related-party Name

An important matching criterion is that the related party (customer or vendor) can be identified from the information on the bank transaction. The related party can be matched to three degrees: Fully, Partially, and None.

The following table shows how the different degrees of related-party match are derived.

Bank account	Name Compared to Related-Party Name	Name Compared to Transaction Text	Related-party Address and/or Related-party City	Unique Name	Result
Exact match	-	-	-	-	Fully
No Match	Exact match (95%)	-	Match	-	Fully
			Match	-	Fully
	Exact match (95%)	-	No Match	No alternatives	Fully
			No Match	Alternative(s) found	Partially
	-	Exact match (95%)	-	No alternatives	Fully
			-	Alternative(s) found	Partially
		Partial match (65%)	-	-	Partially

The following diagram shows the algorithm flow of identifying and matching the related party.



First the algorithm checks if the bank account matches. This is the quickest and most accurate way of identifying a related party. All bank accounts that are assigned to a customer/vendor are checked.

Then the algorithm checks if related-party information on the bank transaction matches with information on any open entries. If a related-party name is defined on the bank transaction, then it must match 95% to be considered a match. To be fully matched, the related-party name must exist only on one open entry or the address or city of the related party must match. If there is related-party information on the bank transaction and the name does not match, then the algorithm does not check the transaction text, because information on the bank transaction is considered the most accurate and because that would

cost in performance.

If there is no related-party name information on the bank transaction, then the algorithm checks the transaction text in case the payer has entered the name manually. In that case, it must match at least 65% to be a partial match. The reason for this is that people often do not enter a full name when making electronic payments. To be a full match, it must match at least 95% match and name must exist only on one open entry. The algorithm does not check for address and city in the transaction text, because it is not considered common that people enter their address in the transaction text when making electronic payments. This rule can easily be changed if needed.

The algorithm uses string nearness for name matching so it can detect if the payer has written the last and first name in a different order or omitted some part of the name. String nearness impacts performance because the function is quite complex and iterates through the entire string if the names do not match, which is the most common case. It is therefore recommended to limit the use of the function.

The result of the related-party matching is set to an instance of the Bank Pmt. Appl. Rule record (table 1252) in the Related Party Matched field.

Match on the Document Number

For a document number to match, the number in the Document No. and/or the External Document No. field on the open entry must be identical to a number found in the transaction text and/or the additional transaction info that the payer has entered.

To increase accuracy, the algorithm does not match on parts of a document number. For higher accuracy, it is recommended to set up the system to use longer document numbers and external document numbers on sales and purchase documents.

The result of the document number matching is set to an instance of Bank Pmt. Appl. Rule record (table 1252) in the Doc. No./Ext. Doc. No. Matched field.

Match on the Amount

An amount can match to three degrees:

- One Match – The amount only exists on one open entry.
- Multiple Matches – The amount exists on multiple open entries.
- No Match – The amount does not exist on any open entries.

Amount matching supports the Payment Tolerance feature by allowing users to set up a payment match tolerance based on percentage or amount on the bank account card. This is implemented as +/- of the amount value.

To match on the amount, the algorithm must also take into account payment discounts. This is performed with the steps:

1. Get the correct amount. Depending on the transaction date, the algorithm gets the value in the Remaining Amount or Remaining Amt. Incl. Discount field in the Bank Statement Matching Buffer table.
2. Check if the transaction amount is within the tolerance range based on the amount gotten in step 1. If it is not, then there is no match.
3. Check if there is one match or multiple matches. First, the algorithm checks how many open entries have payment discounts by setting a filter, Pmt. Discount Due Date < Transaction Date, in the Bank Statement Matching Buffer table and a filter on the amount including the tolerance. Then the algorithm sets the date filter the other way and sets the amount range on the Remaining Amount field (the full amount).
4. The two numbers that result from step 3 are added and the value is shown in the Amount Incl. Tolerance Matched field from an instance of the Bank Pmt. Appl. Rule record (table 1252).

Note: If the payer pays the full amount although a discount was granted, or if the payer pays a discounted amount after the payment discount due date, then there is no match on the amount.

Assign a Match Score

The Bank Pmt. Appl. Rule table defines the match scores and the match confidences. It is implemented as a table to enable the algorithm to be configurable and extensible.

The following levels of match confidence exist:

- High – The application does not need to be reviewed.
- Medium – Good match, but the application should be reviewed. Could be missing amount match.
- Low – Poor match. The application must be reviewed.

The algorithm assigns a match confidence with the following steps:

1. Assign the result values from the Related Party Matched, Doc. No./Ext. Doc. No. Matched, and Amount Incl. Tolerance Matched fields to the parameter row of the Bank Pmt. Appl. Rule table. The green table below shows the parameter row. The blue table shows the configured rules in the Bank Pmt. Appl. Rule table. ((The table is sorted on score descending.)

Match Confidence	Priority	Score	Related Party Matched	Document No. / Ext. Document No. Matched	Number of Entries Within Amount Tolerance Found
			Fully	**Yes**	**One Match**

Match Confidence	Priority	Score	Related Party Matched	Document No. / Ext. Document No. Matched	Number of Entries Within Amount Tolerance Found
High	1	3999	Fully	Yes - Multiple	One Match
High	2	3998	Fully	Yes - Multiple	Multiple Matches
High	3	3997	Fully	Yes	One Match
Medium	1	2999	Fully	Yes - Multiple	Not Considered
Medium	2	2998	Fully	Yes	Not Considered
Medium	3	2997	Fully	No	Multiple Matches
Medium	4	2996	Partially	Yes - Multiple	Not Considered
Low	1	1999	Fully	Yes	No Matches
Low	2	1998	Partially	Yes	No Matches

2. Filter the TempBankPmtApplRule table based on the parameter row. Take the first row (in case multiple parameter rows exist) and assign the match confidence and score from the selected row.

Match Confidence	Priority	Score	Related Party Matched	Document No. / Ext. Document No. Matched	Number of Entries Within Amount Tolerance Found
High		**3997**	**Fully**	**Yes**	**One Match**

Match Confidence	Priority	Score	Related Party Matched	Document No. / Ext. Document No. Matched	Number of Entries Within Amount Tolerance Found
High	3	3997	Fully	Yes	One Match

3. If in *Apply Automatically* mode, the values that can be applied will be added to the match candidate buffer (Bank Statement Matching Buffer table). This step improves performance because the system uses less memory if the values with match confidence None are excluded. Most of the matches would have match confidence None. If in *Propose Entries* mode, insert all values. (There are no performance issues with this mode because the algorithm only performs this step for one bank transaction at a time.)

Text-to-Account Mapping

Users can map text found on recurring payments for which no invoice documents exist, such as fuel expenses and small cash receipts, to specific customer, vendor, or G/L accounts. The mappings are defined in the Text-to-Account Mapping table (1251). The matching algorithm also iterates through these mappings. If a valid text-to-account mapping is found and the match found has a confidence lower than High, then the text-to-account mapping is used. The reason for this prioritization is that if a match of High confidence exists, then it is most likely not due to a text-to-account mapping but reflects the payment of an invoice.

The matching algorithm checks for each entry in the Text-to-Account Mapping table if the value in the Mapping Text field matches with the transaction text on the bank transaction. If it matches, then the pair gets the score 3000, which is one score under the lowest High match confidence, and the value is added to the TempBankStatementMatchingBuffer table.

One-to-Many Matching

It is a common scenario that a payment is made for multiple invoices and that the payer writes the invoice numbers in the transaction text. To support this scenario, the algorithm will match one bank transaction to many open entries if the document numbers are provided.

This is done by building a list of the matched document numbers while doing the one-to-one matching for the bank transaction. A score is then assigned in the same way as for one-to-one matching. If the value in the Document No. or External Document No. field matches, the algorithm inserts or updates a multiple-match line in the TempBankStatementMatchingBuffer table. To distinguish between one-to-one and one-to-many rules, this is indicated in the One-to-Many Match field. Open entries that are matched are tracked in the TempBankStmtMultipleMatchLine variable.

When matching is done, the TempBankStatementMatchingBuffer table contains both one-to-one and one-to-many matches. One-to-many entries get a match score in the same way as for one-to-one matches, by using the TempBankPmtApplRule table. The only difference is that the algorithm always sets the Doc. No./Ext. Doc. No. Matched field in the BankPmtApplRule table to Yes - Multiple when finding the best match. This means that all definitions are in the Bank Pmt. Appl. Rule table and the whole system is configurable.

Default Payment Application Rules

The following table shows the default payment application rules that are provided in the generic version of Microsoft Dynamics NAV. The rules are sorted on match confidence descending, which is the priority order in which the rules are applied by the algorithm.

Match Confidence	Priority	Related Party Matched	Document No./Ext. Document No. Matched	Number of Entries Within Amount Tolerance Found
High	1	Fully	Yes - Multiple	One Match
High	2	Fully	Yes - Multiple	Multiple Matches
High	3	Fully	Yes	One Match
High	4	Fully	Yes	Multiple Matches
High	5	Partially	Yes - Multiple	One Match
High	6	Partially	Yes - Multiple	Multiple Matches
High	7	Partially	Yes	One Match
High	8	Fully	No	One Match
High	9	No	Yes - Multiple	One Match
High	10	No	Yes - Multiple	Multiple Matches
Medium	1	Fully	Yes - Multiple	Not Considered
Medium	2	Fully	Yes	Not Considered
Medium	3	Fully	No	Multiple Matches
Medium	4	Partially	Yes - Multiple	Not Considered
Medium	5	Partially	Yes	Not Considered
Medium	6	No	Yes	One Match
Medium	7	No	Yes-Multiple	Not Considered
Medium	8	Partially	No	One Match
Medium	9	No	Yes	Not Considered
Low	1	Fully	No	No Matches
Low	2	Partially	No	Multiple Matches
Low	3	Partially	No	No Matches
Low	4	No	No	One Match
Low	5	No	No	Multiple Matches

Applying Phase

To apply bank transactions to their matching open entries, the algorithm first sorts the match candidate buffer table descending on score. Then the algorithm goes through all entries in the TempBankStatementMatchingBuffer table and applies them, with the following limitations:

- The bank transaction must not already be applied to another open entry.
- The open entry must not already be applied to another bank transaction.

For one-to-one matching, the algorithm applies the entry number that is specified in TempBankStatementMatchingBuffer table. For one-to-many matching, the algorithm applies the entry number that is specified in the TempBankStmtMultipleMatchLine table according to the apply-oldest-first principle.

The apply-oldest-first principle is needed for one-to-many matching where the payment amount cannot cover all the applied entries, for example because the customer has underpaid. In this case, the algorithm applies to the oldest open entries first and then logs a warning in the Payment Matching Details table, which is displayed in a FactBox.

Matching Example

1. Assume the following data tables. The records are already filtered as eligible, and the customer names are unique:

Bank Acc. Reconciliation Line table

Line No.	Statement Amount	Transaction Text	Payer Information	Additional Transaction Information
10000	-1500	Inv.10001	Cannon Group	Ref. 465754
20000	-1750	Inv.10201	Kennel	No.10201
30000	195	Shell		Ref. 12345
40000	-10000	Inv. 10210, 10211, 10212	Spotsmeyer's Furnishings	

Customer Ledger Entry table

Entry No.	Type	Amount	Document No.	External Doc. No.	Customer No.	Customer Name
1	Invoice	1500	10001	465754	11205	Cannon Group Plc.
2	Invoice	1500	10201	56532	25000	Cardoxy
3	Invoice	750	99876	46575	30000	Kennel
4	Invoice	2500	10210	43243	01121212	Spotsmeyer's Furnishings
5	Invoice	2500	10211	43244	01121212	Spotsmeyer's Furnishings
6	Invoice	5000	10212	43245	01121212	Spotsmeyer's Furnishings

Note: The Customer Name field is not in the table, so it is picked from the Customer table.

Text-to-Account Mapping table

Line No.	Mapping Text	Debit Acc. No.	Credit Account No.	Bal. Source Type	Bal. Source No.
10000	Shell	2610	2610	G/L Account	
20000	Refund private			Customer	10000

2. The match candidate buffer table contains the following records, based on the default payment application rules:

Bank Acc. Rec. Line No.	Entry No.	Related Party Matched	Doc/Ext. Doc. No Matched	Amount	Confidence	Score
10000	1	Fully	Yes	One Match	High	3997
10000	2	No	No	One Match	Low	1996
10000	3	No	Yes	No Match	Medium	2991
20000	1	No	No	No Match	None	0
20000	2	No	Yes	No Match	Medium	2991
20000	3	Fully	No	No Match	Low	1990
30000	-10000	-	-	-	High – Text Mapper	3000
40000	4	Fully	Yes	No match	Medium	2998
40000	5	Fully	Yes	No match	Medium	2998
40000	6	Fully	Yes	No match	Medium	2998
40000	-1 (Matches with the three open entries 4,5,6)	Fully	Multiple	One Match	High	3999

Notes:

- Text-to-account mappings are added with a negative entry number to differentiate them. (30000)
- Non-matching text-to-account mappings are not added to the table.
- If in *Propose Entries* mode, matches with confidence None are not added to the table. (20000)
- One-to-many matches create new entries. Entries for one-to-one matches are present in the table alongside one-to-many matches. One-to-many matches must have a higher score to get applied. (40000)

3. The match candidate buffer table is sorted on the Score field descending. Then the algorithm iterates through each line in the table, which represents a transaction-to-entry pair, and applies the bank transactions to the open entries one by one. If either the transaction or the entry is already applied, then the line is not considered, because that indicates that a higher score assigned earlier. An exception to this principle is that text-to-account mappings can be used several times but only to bank transactions that have not been applied before.

Bank Acc. Rec. Line No.	Entry No.	Related Party Matched	Doc/Ext. Doc. No Matched	Amount	Confidence	Score	Application Result
40000	-1 (Matches with the three open entries 4,5,6)	Fully	Multiple	One Match	High	3999	Applied to entries 4,5 and 6
10000	1	Fully	Yes	One Match	High	3997	Applied to entry 1
40000	4	Fully	Yes	No match	Medium	2998	Not applied - Ledger entry and Line already applied
40000	5	Fully	Yes	No match	Medium	2998	Not applied - Ledger entry and Line already applied
40000	6	Fully	Yes	No match	Medium	2998	Not applied - Ledger entry and Line already applied
10000	3	No	Yes	No Match	Medium	2991	Not applied - Line already applied
20000	2	No	Yes	No Match	Medium	2991	Applied to entry 2
10000	2	No	No	One Match	Low	1996	Not applied since line is already applied
20000	1	No	No	No Match	None	0	Not applied - Ledger entry and Line already applied
20000	3	Fully	No	No Match	Low	1990	Not applied – line already applied
30000	-10000	-	-	-	High – Text Mapper	3000	Applied to text mapper rule

Extending the Algorithm

Modifying Payment Application Rules

The first way to extend the matching algorithm is to add, modify, or remove rules in the Bank Pmt. Appl. table. You can access this table in the UI through the Payment Application Rules window.

The following table shows an example of two new rules for when the related-party information exists.

Match Confidence	Priority	Related Party Matched	Document No. / Ext. Document No. Matched	Number of Entries Within Amount Tolerance Found
High	1	Fully	Yes - Multiple	One Match
High	2	Fully	Yes - Multiple	Multiple Matches
High	3	Fully	Yes	One Match
Medium	1	Fully	Yes - Multiple	Not Considered
Medium	2	Fully	Yes	Not Considered
Medium	3	Fully	No	Multiple Matches
Medium	4	Partially	Yes - Multiple	Not Considered
Low	1	Fully	Yes	No Matches
Low	2	Partially	Yes	No Matches
Low	3	Fully	No	No Matches
Low	4	Partially	No	One Match

To implement the modification in every new company and upgraded databases, modify the InsertDefaultMatchingRules method in the Bank Pmt. Appl. Rule table.

Adding Additional Matching Criteria

The second way to extend the matching algorithm is to add additional matching criteria. This is done by adding new columns in the Bank Pmt. Appl. Rule tale. (Adding columns to the table is good for upgrade since it will not cause merge issues.)

The following table shows an example of a new matching criterion for the document date.

Match Confidence	Priority	Related Party Matched	Document No. / Ext. Document No. Matched	Number of Entries Within Amount Tolerance Found	Document Date
High	1	Fully	Yes - Multiple	One Match	
High	2	Fully	Yes - Multiple	Multiple Matches	
High	3	Fully	Yes	One Match	
Medium	1	Fully	Yes - Multiple	Not Considered	
Medium	2	Fully	Yes	Not Considered	
Medium	3	Fully	No	Multiple Matches	
Medium	4	Partially	Yes - Multiple	Not Considered	**Overdue**
Low	1	Fully	Yes	No Matches	
Low	2	Partially	Yes	No Matches	
Low	3	Fully	No	No Matches	**Overdue**
Low	4	Partially	No	One Match	**Overdue**

In the new column, the code must be updated in two places:

1. Update the FindMatchingEntry function in the Match Bank Payments codeunit (1255)
2. Update the GetBestMatchScore function in Bank Pmt. Appl. Rule table (1252).
3. Update the FindMatchingEntry function in codeunit 1255 and the GetBestMatchScore function in table 1252, as show in green font below.

```
LOCAL FindMatchingEntry(TempLedgerEntryMatchingBuffer : TEMPORARY Record "Ledger Entry Matching Buffer";VAR BankAccReconciliatio
IF CanEntriesMatch(
     BankAccReconciliationLine,TempLedgerEntryMatchingBuffer."Remaining Amount",TempLedgerEntryMatchingBuffer."Posting Date")
THEN BEGIN
  RelatedPartyMatching(BankPmtApplRule,TempLedgerEntryMatchingBuffer,BankAccReconciliationLine,AccountType);
  DocumentMatching(BankPmtApplRule,BankAccReconciliationLine,
    TempLedgerEntryMatchingBuffer."Document No.",TempLedgerEntryMatchingBuffer."External Document No.");

  RemainingAmount := TempLedgerEntryMatchingBuffer.GetApplicableRemainingAmount(BankAccReconciliationLine,UsePaymentDiscounts);
  AmountInclToleranceMatching(
    BankPmtApplRule,BankAccReconciliationLine,AccountType,RemainingAmount);

  //DocumentIsOverdue(BankPmtApplRule,TempLedgerEntryMatchingBuffer,BankAccReconciliationLine,AccountType);
  Score := TempBankPmtApplRule.GetBestMatchScore(BankPmtApplRule); // Modify this function as well
```

```
GetBestMatchScore(ParameterBankPmtApplRule : Record "Bank Pmt. Appl. Rule") : Integer
CLEAR(Rec);
SETCURRENTKEY(Score);
ASCENDING(FALSE);

SETFILTER("Related Party Matched",'%1|%2',
  ParameterBankPmtApplRule."Related Party Matched",
  ParameterBankPmtApplRule."Related Party Matched"::"Not Considered");

SETFILTER("Doc. No./Ext. Doc. No. Matched",'%1|%2',
  ParameterBankPmtApplRule."Doc. No./Ext. Doc. No. Matched",
  ParameterBankPmtApplRule."Doc. No./Ext. Doc. No. Matched"::"Not Considered");

SETFILTER("Amount Incl. Tolerance Matched",'%1|%2',
  ParameterBankPmtApplRule."Amount Incl. Tolerance Matched",
  ParameterBankPmtApplRule."Amount Incl. Tolerance Matched"::"Not Considered");

// Add new filter for OverDue

IF FINDFIRST THEN
  EXIT(Score);

EXIT(0);
```

Note: The Not Considered option can be used to keep the rules list short by combining multiple rules with the same match confidence. For example, to add two rules for confidence Medium when the relayed-party is partially matched and the document number is matched, the Amount Incl. Tolerance Matched field can be set to Not Considered. The alternative without this option would be to create two separate rules with the Amount Incl. Tolerance Matched field set to Yes and No respectively.

Adding New Option Values

All parameter fields are added as options in the Bank Pmt. Appl. Rule table. If needed, option values can be added, and the table can be extended with new rules. To include the new option values, the functions that set current parameter must be updated. (During development of Microsoft Dynamics NAV 2015, the algorithm was extended with one-to-many matching by adding the Doc. No./Ext. Doc. No. Matched::Multiple option.)

Including More Information from Entries

To include more information from the open customer or vendor leger entries, the fields in question can be added to the Ledger Entry Matching Buffer table (1248). Optionally, the fields can be added to the Bank Statement Matching Buffer table (1250).

Performance

Considerations

An important performance consideration is that the maximum time that it takes to open a page should be 10s. If it takes longer than that, the user feels interrupted and will start doing something else.

The highest impact on the performance of automatic record matching is the number of open entries and the number of bank transactions per file. The second-highest impact is the method by which a related party is identified.

To improve the performance of a high-volume installation, it is recommended that the related-party identification step is modified first since this will increase performance the most. This can be done, for example, by removing or disabling identification methods that are not applicable.

Measurements

Based on feedback, Microsoft Dynamics NAV 2015 uses the following data sizes for performance measurement for two critical actions, running the Apply Automatically function and opening the Payment Application window.

Number of Open Ledger Entries	Number of Customers	Bank Transactions per Bank Statement File	Time to Apply Automatically	Time to Open Payment Application Window
500	5000	100	10s	2s
1000	10000	250	1 min	3s
5000	20000	500	9 min	5s
10000	50000	500	16 min	7s
10000	50000	1000	34 min	7s
20000	50000	500	34 min	14s
20000	50000	1000	139 min	14s

From the measurements we can see that the number of customers in the system does not impact the performance. It is the number of open ledger entries and the number of bank transactions per statement file that impact performance.

To optimize for high-volume scenarios, the algorithm must be modified. This depends on the process in the company. The places where the algorithm could potentially be optimized are:

1. Remove the identification of the related party on data that is not available.
2. If the related party is identified with a high match confidence for a given line, then do not check the open ledger entries for other related parties.

Conclusion

This white paper explained how automatic record matching is performed in the Payment Reconciliation Journal window where bank transactions are matched with open customer and vendor ledger entries that the related payments can be applied to.

+ Share

Microsoft Dynamics NAV