

# **Retek<sup>®</sup> Invoice Matching<sup>™</sup> 11.0.2**

## **Operations Guide Addendum**



---

**Corporate Headquarters:**

Retek Inc.  
Retek on the Mall  
950 Nicollet Mall  
Minneapolis, MN 55403  
USA  
888.61.RETEK (toll free US)  
Switchboard:  
+1 612 587 5000  
Fax:  
+1 612 587 5100

**European Headquarters:**

Retek  
110 Wigmore Street  
London  
W1U 3RW  
United Kingdom  
Switchboard:  
+44 (0)20 7563 4600  
Sales Enquiries:  
+44 (0)20 7563 46 46  
Fax:  
+44 (0)20 7563 46 10

The software described in this documentation is furnished under a license agreement, is the confidential information of Retek Inc., and may be used only in accordance with the terms of the agreement.

No part of this documentation may be reproduced or transmitted in any form or by any means without the express written permission of Retek Inc., Retek on the Mall, 950 Nicollet Mall, Minneapolis, MN 55403, and the copyright notice may not be removed without the consent of Retek Inc.

Information in this documentation is subject to change without notice.

Retek provides product documentation in a read-only-format to ensure content integrity. Retek Customer Support cannot support documentation that has been changed without Retek authorization.

The functionality described herein applies to this version, as reflected on the title page of this document, and to no other versions of software, including without limitation subsequent releases of the same software component. The functionality described herein will change from time to time with the release of new versions of software and Retek reserves the right to make such modifications at its absolute discretion.

Retek<sup>®</sup> Invoice Matching<sup>™</sup> is a trademark of Retek Inc.

Retek and the Retek logo are registered trademarks of Retek Inc.

This unpublished work is protected by confidentiality agreement, and by trade secret, copyright, and other laws. In the event of publication, the following notice shall apply:

©2004 Retek Inc. All rights reserved.

All other product names mentioned are trademarks or registered trademarks of their respective owners and should be treated as such.

Printed in the United States of America.

## Customer Support

### Customer Support hours

Customer Support is available 7x24x365 via email, phone, and Web access.

Depending on the Support option chosen by a particular client (Standard, Plus, or Premium), the times that certain services are delivered may be restricted. Severity 1 (Critical) issues are addressed on a 7x24 basis and receive continuous attention until resolved, for all clients on active maintenance. Retek customers on active maintenance agreements may contact a global Customer Support representative in accordance with contract terms in one of the following ways.

### Contact Method    Contact Information

**E-mail**                      support@retex.com

**Internet (ROCS)**    [rocs.retek.com](https://rocs.retek.com)  
Retek's secure client Web site to update and view issues

**Phone**                      +1 612 587 5800

Toll free alternatives are also available in various regions of the world:

Australia	+1 800 555 923 (AU-Telstra) or +1 800 000 562 (AU-Optus)
France	0800 90 91 66
Hong Kong	800 96 4262
Korea	00 308 13 1342
United Kingdom	0800 917 2863
United States	+1 800 61 RETEK or 800 617 3835

**Mail**                      Retek Customer Support  
Retek on the Mall  
950 Nicollet Mall  
Minneapolis, MN 55403

### When contacting Customer Support, please provide:

- Product version and program/module name.
- Functional and technical description of the problem (include business impact).
- Detailed step-by-step instructions to recreate.
- Exact error message received.
- Screen shots of each step you take.

# Contents

<b>Chapter 1 – RETL program overview for the ReIM extraction program .....</b>	<b>1</b>
Architectural design .....	1
ReIM extraction architecture .....	2
Configuration .....	2
RETL .....	2
RETL user and permissions .....	2
Environment variables .....	2
dwi_config.env settings .....	3
Program features .....	4
Program status control files .....	4
Restart and recovery .....	5
Bookmark file .....	5
Message logging .....	5
Daily log file .....	6
Format .....	6
Program error file .....	6
ReIME reject files .....	7
Schema files .....	7
Resource files .....	7
Command line parameters .....	8
Typical run and debugging situations .....	8
RETL extraction program list .....	9
RETL extract program flow diagram .....	10
Legend .....	10
Program flow diagram .....	10
Application programming interface (API) flat file specifications .....	10
API format .....	11
File layout .....	11
General business rules and standards common to all APIs .....	11
sincilddm.txt .....	13



# Chapter 1 – RETL program overview for the ReIM extraction program

To facilitate the extraction of data from ReIM (that could be eventually loaded into a data warehouse for reporting purposes, for example), ReIM works in conjunction with the Retek Extract Transform and Load (RETL) framework. This architecture optimizes a high performance data processing tool that can let database batch processes take advantage of parallel processing capabilities.

Retek's streamlined RETL code provides for less data storage, easier implementation, and reduced maintenance requirements through decreased code volume and complexity. The RETL scripts are Korn shell scripts that are executable from a Unix prompt. A typical run and debugging situation is provided later in this chapter.

These extractions were initially designed for Retek Data Warehouse (RDW) but can be used for some other application in the retailer's enterprise.

For more information about the RETL tool, see the latest RETL Programmer's Guide.

## Architectural design

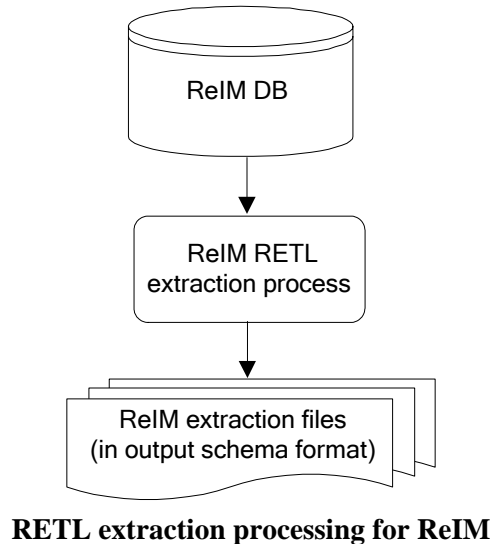
The diagram below illustrates the extraction processing architecture for ReIM. Instead of managing the change captures as they occur in the source system during the day, the process involves extracting the current data from the source system. The extracted data is output to flat files. These flat files are then available for consumption by a product such as Retek Data Warehouse (RDW).

The target system, (RDW, for example), has its own way of completing the transformations and loading the necessary data into its system, where it can be used for further processing in the environment.

ReIM modules use the same libraries, resource files and configuration files as RMS. All these libraries, resource files and configure files are packed with RMS. ReIM must have RMS installed before any ReIM RETL scripts can be 'kicked off'.

### ReIM extraction architecture

The architecture relies upon the use of well-defined flows specific to the ReIM database. The resulting output is comprised of data files written in a well-defined schema file format. This extraction includes no destination specific code.



## Configuration

### RETL

Before trying to configure and run ReIM ETL, install RETL version 11.2 or later, which is required to run ReIM 11.0.2 RETL. Run the 'verify\_retl' script (included as part of the RETL installation) to ensure that RETL is working properly before proceeding.

### RETL user and permissions

ReIM ETL is installed and run as the RETL user. Additionally, the permissions are set up as per the RETL Programmer's Guide. ReIM ETL reads data, creates, deletes and updates tables. If these permissions are not set up properly, extractions fail.

### Environment variables

See the RETL Programmer's Guide for RETL environment variables that must be set up for your version of RETL. You will need to set MMHOME to your base directory for ReIM RETL. This is the top level directory that you selected during the installation process. In your .kshrc, you should add a line such as the following:

```
export MMHOME=<base directory for RMS ETL>\dwi11.0\dev
```



**Note:** Because ReIM modules share the same libraries and configuration files as RMS, MMHOME is the same as what is defined in RMS.



### dwi\_config.env settings

Make sure to review the environmental parameters in the dwi\_config.env file before executing batch modules. There are several variables you must change depending upon your local settings:

For example:

```
export DBNAME=int9i
export RIM_OWNER=steffej_reim1102
export BA_OWNER=rmsint1102
export ORACLE_PORT="1524"
export ORACLE_HOST="mspdev38"
```

You must set up the environment variable PASSWORD in dwi\_config.env. In the example below, adding the line to the dwi\_config.env causes the password 'mypasswd' to be used to log into the database:

```
export PASSWORD=mypasswd
```

### Steps to configure RETL

- 1 Log in to the Unix server with a Unix account that will run the RETL scripts.
- 2 Change directories to \$MMHOME/rfx/etc.
- 3 Modify the dwi\_config.env script:
  - a Change the DBNAME variable to the name of the ReIM database.
  - b Change the RIM\_OWNER variable to the username of the ReIM schema owner.
  - c Change the BA\_OWNER variable to the username of the ReIME batch user.
  - d Change the ORACLE\_HOST variable to the database server name.
  - e Change the ORACLE\_PORT variable to the database port number
  - f Change the MAX\_NUM\_COLS variable to modify the maximum number of columns from which RETL selects records.



**Note:** All ReIM tables must be under the RMS database. ReIM has the same BA\_OWNER as RMS. Thus, the only piece that ReIM modifies in dwi\_config.env file is to assign a value to RIM\_OWNER. The configuration file, dwi\_config.env, as well as all other configuration files, are packed with RMS.

# Program features

RETL programs use one return code to indicate successful completion. If the program successfully runs, a zero (0) is returned. If the program fails, a non-zero is returned.

## Program status control files

To prevent a program from running while the same program is already running against the same set of data, the ReIME code utilizes a program status control file. At the beginning of each module, `dwi_config.env` is run. It checks for the existence of the program status control file. If the file exists, then a message stating, ‘`{PROGRAM_NAME}` has already started’, is logged and the module exits. If the file does not exist, a program status control file is created and the module executes.

If the module fails at any point, the program status control file is not removed, and the user is responsible for removing the control file before re-running the module.

## File naming conventions

The naming convention of the program status control file allows a program whose input is a text file to be run multiple times at the same time against different files.

The name and directory of the program status control file is set in the configuration file (`dwi_config.env`). The directory defaults to `$MMHOME/error`. The naming convention for the program status control file itself defaults to the following dot separated file name:

- The program name
- The first filename, if one is specified on the command line
- ‘status’
- The business virtual date for which the module was run

For example, the program status control file for the `invildex` program would be named as follows for the `VDATE` of March 21, 2004:

```
$MMHOME/error/sincildex.sincilddm.txt.status.20040321
```

### Restart and recovery

Because RETL processes all records as a set, as opposed to one record at a time, the method for restart and recovery must be different from the method that is used for Pro\*C. The restart and recovery process serves the following two purposes:

- 1 It prevents the loss of data due to program or database failure.
- 2 It increases performance when restarting after a program or database failure by limiting the amount of reprocessing that needs to occur.

The ReIM extract module (ReIME) extracts from a source transaction database writes to a text file.

To limit the amount of data that needs to be re-processed, more complicated modules that require the use of multiple RETL flows utilize a bookmark method for restart and recovery. This method allows the module to be restarted at the point of last success and run to completion. The bookmark restart/recovery method incorporates the use of a bookmark flag to indicate which step of the process should be run next. For each step in the process, the bookmark flag is written to and read from a bookmark file.



**Note:** If the fix for the problem causing the failure requires changing data in the source table or file, then the bookmark file must be removed and the process must be re-run from the beginning in order to extract the changed data.

### Bookmark file

The name and directory of the restart and recovery bookmark file is set in the configuration file (dwi\_config.env). The directory defaults to \$MMHOME/rfx/bookmark. The naming convention for the bookmark file itself defaults to the following 'dot'-separated file name:

- The program name
- The first filename, if one is specified on the command line
- 'bkm'
- The business virtual date for which the module was run

The example below illustrates the bookmark flag for the invildex program run on the VDATE of January 5, 2004:

```
$MMHOME/rfx/bookmark/sincildex.sincilddm.txt.bkm.20040105
```

### Message logging

Message logs are written daily in a format described in this section.

### Daily log file

Every RETL program writes a message to the daily log file when it starts and when it finishes. The name and directory of the daily log file is set in the configuration file (dwi\_config.env). The directory defaults to \$MMHOME/log. All log files are encoded UTF-8.

The naming convention of the daily log file defaults to the following 'dot' separated file name:

- The business virtual date for which the modules are run
- '.log'

For example, the location and the name of the log file for the business virtual date (VDATE) of March 21, 2004 would be the following:

```
$MMHOME/log/20040321.log
```

### Format

As the following examples illustrate, every message written to a log file has the name of the program, a timestamp, and either an informational or error message:

```
sincildex 12:51:07: Program starting...
sincildex 12:51:07: last max post date is 20010311000000
sincildex 12:51:07: Retrieve current max post date
sincildex 12:51:10: Loading invc_exchng_rate_temp table ...
sincildex 12:51:15: Loading po_exchng_rate_temp table ...
sincildex 12:51:20: Process all records between last post date and current
max post date
sincildex 12:51:27: Drop table rmsint110buser1.INVC_EXCHNG_RATE_TEMP
sincildex 12:51:27: Drop table rmsint110buser1.PO_EXCHNG_RATE_TEMP
sincildex 12:51:27: Number of records in sincilddm.txt = 15
sincildex 12:51:27: Program completed successfully
```

If a program finishes unsuccessfully, an error file is usually written that indicates where the problem occurred in the process. There are some error messages written to the log file, such as 'No output file specified', that require no further explanation written to the error file.

### Program error file

In addition to the daily log file, each program also writes its own detail flow and error messages. Rather than clutter the daily log file with these messages, each program writes out its errors to a separate error file unique to each execution.

The name and directory of the program error file is set in the configuration file (dwi\_config.env). The directory defaults to \$MMHOME/error. All errors and *all routine processing messages* for a given program on a given day go into this error file (for example, it will contain both the stderr and stdout from the call to RETL). All error files are encoded UTF-8.

The naming convention for the program's error file defaults to the following 'dot' separated file name:

- The program name
- The first filename, if one is specified on the command line
- The business virtual date for which the module was run

For example, all errors and detail log information for the invildex program would be placed in the following file for the batch run of March 21, 2004:

```
$MMHOME/error/sincildex.sincilddm.txt.20040321
```

### ReIME reject files

The ReIME extract module may produce a reject file if it encounters data related problems, such as an inability to find data on required lookup tables. The module tries to process all data and then indicates that records were rejected so that all data problems can be identified in one pass and corrected; then, the module can be re-run to successful completion. If a module does reject records, the reject file is *not* removed, and the user is responsible for removing the reject file before re-running the module.

The records in the reject file contain an error message and key information from the rejected record. The following example illustrates a record that is rejected due to problems within the currency conversion library:

```
Unable to convert currency for LOC_IDNT, DAY_DT|3|20011002
```

The name and directory of the reject file is set in the configuration file (dwi\_config.env). The directory defaults to \$MMHOME/data.



**Note:** A directory specific to reject files can be created. The dwi\_config.env file would need to be changed to point to that directory.

### Schema files

RETL uses schema files to specify the format of incoming or outgoing datasets. The schema file defines each column's data type and format, which is then used within RETL to format/handle the data. For more information about schema files, see the latest RETL Programmer's Guide. Schema file names are hard-coded within each module since they do not change on a day-to-day basis. All schema files end with ".schema" and are placed in the "rfx/schema" directory.

### Resource files

The ReIM Kornshell program uses resource files so that the same RETL program can run in various language environments. For each language, there is one resource file.

Resource files contain hard-coded strings that are used by extract programs. The name and directory of the resource file is set in the configuration file (dwi\_config.env). The default directory is \${MMHOME}/rfx/include.

The naming convention for the resource file follows the two-letter ISO code standard abbreviation for languages (for example, en for English, fr for French, ja for Japanese, es for Spanish, de for German, and so on).



**Note:** Resource files are packed only with RMS.

### Command line parameters

The module handles command line parameters in the way described in this section. See the section, ‘RETL extraction program list’ to determine the command line parameters for a module.



**Note:** For some RETL modules across Retek products, default output file names and schema names correspond to RDW program names.

#### A non-file based module that requires parameters

In order for the non-file based RETL module to run, command line parameters need to be passed in at the Unix command line. This ReIME module requires an `output_file_path` and `output_file_name` to be passed in. This module may allow the operator to specify more than one output file.

For example:

```
sincildex.ksh output_file_path/output_file_name
```

### Typical run and debugging situations

The following examples illustrate typical run and debugging situations for types of programs. The log, error, and so on file names referenced below assume that the module is run on the business virtual date of March 9, 2004. See the previously described naming conventions for the location of each file.

For example:

To run `sincildex.ksh`:

- 1 Change directories to `$MMHOME/rfx/src`.
- 2 At a Unix prompt enter:  

```
%sincildex.ksh $MMHOME/data/sincilddm.txt
```

If the module runs successfully, the following results:

- 1 **Log file:** Today’s log file, `20040309.log`, contains the messages “Program started ...” and “Program completed successfully” for `sincildex.ksh`.
- 2 **Data:** The `sincilddm.txt` file exists in the `$MMHOME/data` directory and contains the extracted records.
- 3 **Error file:** The program’s error file, `sincildex.sincilddm.txt.20040309`, contains the standard RETL flow (ending with “All threads complete” and “Flow ran successfully”) and no additional error messages.
- 4 **Program status control:** The program status control file, `sincildex.sincilddm.txt.status.20040309`, does not exist.
- 5 **Reject file:** The reject file, `sincildex.sincilddm.txt.rej.20040309`, does not exist.

If the module does *not* run successfully, the following results:

- 1 **Log file:** Today's log file, 20040309.log, does not contain the "Program completed successfully" message for sincilddm.ksh.
- 2 **Data:** The sincilddm.txt file may exist in the data directory but may not contain all the extracted records.
- 3 **Error file:** The program's error file, sincilddm.txt.20040309, may contain an error message.
- 4 **Program status control:** The program status control file, sincilddm.txt.status.20040309, exists.
- 5 **Reject file:** The reject file, sincilddm.txt.rej.20040309, does not exist because this module does not reject records.
- 6 **Bookmark file (in certain conditions):** The bookmark file, sincilddm.txt.bkm.20040309, exists because this module contains more than one flow. The error occurred after the first flow (for example, during the second flow).

To re-run a module from the beginning, perform the following actions:

- 1 Determine and fix the problem causing the error.
- 2 Remove the program's status control file.
- 3 Remove the bookmark file from \$MMHOME/rfx/bookmark
- 4 Change directories to \$MMHOME/rfx/src. At a Unix prompt, enter:  

```
%sincilddm.ksh $MMHOME/data/sincilddm.txt
```



**Note:** To understand how to engage in the restart and recovery process, see the section, 'Restart and recovery' earlier in this chapter.

## RETL extraction program list

This section serves as a reference to the RETL extraction ReIM program.

Program	Functional Area	Source Table or File	Schema File	Target File	Arguments
sincilddm.ksh	Supplier Invoice Cost	IM_DOC_HEAD, IM_INVOICE_DE TAIL, ORDLOC, ITEM_MASTER	sincilddm.sche ma	sincilddm.txt	output_file_ path/filenam e

## RETL extract program flow diagram

This section presents the flow diagram for data processing. The source system's program is illustrated along with the program or process that interfaces with the source.

Before setting up a program schedule, familiarize yourself with the functional and technical constraints associated with each program.

### Legend



**Note:** See the RMS Operations Guide for more information regarding the modules shown in the legend below.



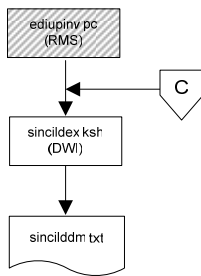
RMS module on which RETL extract modules are dependent



signifies the completion of pre-DWI maintenance module to create currency conversion tables

### Program flow diagram

Supplier invoice  
cost



## Application programming interface (API) flat file specifications

This section contains APIs that describe the file format specifications for all text files.

In addition to providing individual field description and formatting information, the APIs provide basic business rules for the incoming data.



## API format

Each API contains a business rules section and a file layout. Some general business rules and standards are common to all APIs. The business rules are used to ensure the integrity of the information held within RDW. In addition, each API contains a list of rules that are specific to that particular API.

### File layout

- **Field Name:** Provides the name of the field in the text file.
- **Description:** Provides a brief explanation of the information held in the field.
- **Data Type/Bytes:** Includes both data type and maximum column length. Data type identifies one of three valid data types: character, number, or date. Bytes identifies the maximum bytes available for a field. A field may not exceed the maximum number of bytes (note that ASCII characters usually have a ratio of 1 byte = 1 character)
  - **Character:** Can hold letters (a,b,c...), numbers (1,2,3...), and special characters (\$,#,&...)
  - **Numbers:** Can hold only numbers (1,2,3...)
  - **Date:** Holds a specific year, month, day combination. The format is “YYYYMMDD”, unless otherwise specified.
- **Any required formatting for a field is conveyed in the Bytes section.** For example, Number(18,4) refers to number precision and scale. The first value is the precision and always matches the maximum number of digits for that field; the second value is the scale and specifies, of the total digits in the field, how many digits exist to the right of the decimal point. For example, the number –12345678901234.1234 would take up twenty ASCII characters in the flat file; however, the overall precision of the number is still (18,4).
- **Field Order:** Identifies the order of the field in the schema file.
- **Required Field:** Identifies whether the field can hold a null value. This section holds either a ‘yes’ or a ‘no’. A ‘yes’ signifies the field may not hold a null value. A ‘no’ signifies that the field may, but is not required, to hold a null value.

### General business rules and standards common to all APIs

- **Complete ‘snapshot’ (of what RDW refers to as dimension data):**  
A majority of RDW’s dimension code requires a complete view of all current dimensional data (regardless of whether the dimension information has changed) once at the end of every business day. If a complete view of the dimensional data is not provided in the text file, invalid or incorrect dimensional data can result. For instance, not including an active item in the prditmdm.txt file causes that item to be closed (as of the extract date) in the data warehouse. When a sale for the item is processed, the fact program will not find a matching ‘active’ dimension record. Therefore, it is essential, unless otherwise noted in each API’s specific business rules section, that a complete snapshot of the dimensional data be provided in each text file.

If there are no records for the day, an empty flat file must still be provided.

- Updated and new records of (what RDW refers to as fact data):  
Facts being loaded to RDW can either be new or updated facts. Unlike dimension snapshots, fact flat files will only contain new/updated facts exported from the source system once per day (or week, in some cases). Refer to each API's specific business rules section for more details.

If there are no new or changed records for the day, an empty flat file must still be provided.

- Primary and local currency amount fields  
Amounts will be stored in both primary and local currencies for most fact tables. If the source system uses multi-currency, then the primary currency column holds the primary currency amount, and the local currency column holds the local currency amount. If the location happens to use the primary currency, then both primary and local amounts hold the primary currency amount. If the source system does not use multi-currency, then only the primary currency fields are populated and the local fields hold NULL values.
- Leading/trailing values:  
Values entered into the text files are the exact values processed and loaded into the datamart tables. Therefore, the values with leading and/or trailing zeros, characters, or nulls are processed as such. RDW does not strip any of these leading or trailing values, unless otherwise noted in the individual API's business rules section.
- Indicator columns:  
Indicator columns are assumed to hold one of two values, either "Y" for yes or "N" for no.
- Delimiters:



**Note:** Make sure the delimiter is never part of your data.

- Dimension Flat File Delimiter Standards (as defined by RDW): Within dimension text files, each field must be separated by a pipe ( | ) character, for example a record from prddivdm.txt may look like the following:

```
1000|1|Homewares|2006|Henry Stubbs|2302|Craig Swanson
```

- Fact Flat File Delimiter Standards (as defined by RDW): Within facts text files, each field must be separated by a semi-colon character ( ; ). For example a record from exchngratedm.txt may look like the following:

```
WIS;20010311;1.73527820592648544918
```

See the latest RETL Programmer's Guide for additional information.

- End of Record Carriage Return:  
Each record in the text file must be separated by an end of line carriage return. For example, the three records below, in which each record holds four values, should be entered as:

```
1|2|3|4
5|6|7|8
9|10|11|12
```

and not as a continuous string of data, such as:

```
1|2|3|4|5|6|7|8|9|10|11|12
```

## sincilddm.txt

Business rules:

- This interface file contains invoice and order cost information for each item on a matched invoice.
- This interface file cannot contain duplicate transactions for an item\_idnt, po\_idnt, invc\_idnt, supp\_idnt, day\_dt, and loc\_idnt combination.
- This interface file follows the fact flat file interface layout standard.
- This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

Name	Description	Data Type/Bytes	Field order	Required field
ITEM_IDNT	The unique identifier of an item.	CHARACTER(25)	1	Yes
PO_IDNT	The unique identifier of a purchase order	VARCHAR2(8)	2	Yes
INVC_IDNT	The unique identifier of an invoice.	VARCHAR2(10)	3	Yes
SUPP_IDNT	The unique identifier of a supplier.	CHARACTER(10)	4	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	5	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	6	Yes
F_SUPP_INVC_UNIT_COST_AMT	The invoice cost, in the system primary currency.	NUMBER(18,4)	7	No
F_SUPP_INVC_UNIT_COST_AMT_LCL	The invoice cost, in the local currency	NUMBER(18,4)	8	No
F_SUPP_INVC_QTY	The quantity of an item shown on the invoice	NUMBER(12,4)	9	No

## Retek Invoice Matching

---

Name	Description	Data Type/Bytes	Field order	Required field
F_PO_ITEM_UNIT_COST_AMT	The item's purchase order unit cost, in primary currency.	NUMBER(18,4)	10	No
F_PO_ITEM_UNIT_COST_AMT_LCL	The item's purchase order unit cost, in local currency.	NUMBER(18,4)	11	No