M-Port Integrator Manual

1 Welcome

Welcome to RedAnt M-Port!

M-Port is a comprehensive middleware solution for Mamut Business Software. It is able to perform actions on data ranging from XML and CSV formats to PDF, and store files, send e-mails, and print reports. Using web services or FTP, it allows you to attach Mamut to any other platform that accepts incoming data in XML or CSV format. In short, it is Mamut's gateway to the outside world.

Download location: http://software.redant.nl/m-port

When using a Crystal Reports workflow, the CR redistributable must be installed from: http://software.redant.nl/cr2008.msi

2 Workflows

Generally speaking, the inner workings of M-Port are determined by installed workflows - different workflows may mean completely different behaviours for M-Port. A workflow is a description of a data path that specifies input, data manipulation or selection processors, and output format and location. M-Port is designed to give access to multiple workflows, but will only execute one workflow at a time.

Workflows are defined in XML format, and consist of the following sections:

- Workflow configuration;
- Origin connection;
- Input format;
- Data processors (optionally);
- Output format;
- Destination connection.

When executed, a workflow follows the steps above closely:

- it makes a connection to the origin;
- reads in data from the origin connection specified in the input format into an intermediate format;
- optionally, manipulates the data using the specified processors;
- transforms the data in intermediate format to the format specified by the output format;
- makes a connection to the destination ('making a connection' may consist of opening a file for writing);
- and writes the formatted data to the destination connection.

So, both the origin and destination sections contain connection settings. These will be outlined in the chapter Connections. Both the input and output sections describe formats, outlined in a dedicated chapter called Formats. In this manual, you will also find a full description of the intermediate format's data definition.

2.1 Configuration

In the workflow's configuration section, the following nodes can be used. Unless noted otherwise, all nodes are required.

node name	description
`customer`	The name of the customer. This is used in the reseller's interface for customer workflows.
`name`	A name that describes the workflow's purpose as condensed as possible. The workflow's name is displayed in the interface menu *Workflows*.
`description` *(optional)*	The workflow's description is visible through the *Info* button in M-Port, and can be used to give the customer more information on the process.
`author` *(optional)*	The workflow's author and contact information.
`datatype`	The central datatype of the workflow. This can be, for instance, *order* or *contact*, for importing or exporting orders or contacts, respectively.
`locale` *(optional)*	The workflow's locale determines decimal points (. or ,) and date formatting. Use, for instance, *nl* or *en*.

2.2 Logging and Backups

M-Port stores a copy of all files it successfully imports and exports using FTP in a dedicated backup directory, as well as all files transferred by e-mail and files removed from their original location after an import. In this way, files read or created by M-Port can be investigated if needed.

2.3 Datastore location

M-Port, by default, stores its workflows in a machine-dependent storage directory. That means all users of the application on one single machine, in principal, share the same workflows.

This setting is contained within the key `HKEY_CURRENT_USER\Software\RedAnt\RedAnt M-Port`, which is created when the application is started for the first time. Using the *Registry Editor* (*Start* > *Run* > `regedit`), you can add a new String value to that key, which is called `DatastoreLocation`. M-Port will then use the specified path instead of the default location.

In addition, a machine-wide setting can be set using the key `HKEY_LOCAL_MACHINE\Software\RedAnt\RedAnt M-Port`'s String value `DatastoreLocation`, which takes precedence over the current user key mentioned above. This key is not created automatically. On 64-bit systems, this key is located at `HKEY_LOCAL_MACHINE\Software\Wow6432Node\RedAnt\RedAnt M-Port`.

You need to restart M-Port in order to effectuate changes in the datastore location.

Default datastore locations

operating system	location
Windows XP Windows Server 2003	`C:\Documents and Settings\All Users\Application Data\RedAnt\M-Port`
Windows Vista Windows 7 Windows Server 2008	`C:\ProgramData\RedAnt\M-Port`

The entire folder needs to be copied for back-up purposes.

2.4 Multiple Companies on One Machine

By default, all M-Port users on one single machine share the same workflows. If this functionality is not desired, for any reason, you can use the `DatastoreLocation` key mentioned above. Another approach can be taken to limit the visible workflows for any particular customer.

To show only applicable workflows, browse to the `HKEY_CURRENT_USER\Software\RedAnt\RedAnt M-Port` key in the *Registry Editor* and create a new String value called *Customer*. When created, M-Port limits visible workflows to only those which have the *exact* same customer name in the `customer`-node.

You need to restart M-Port in order to effectuate changes in the Customer name.

2.5 Debug Mode

To allow easy access to M-Port's isolated storage, you can enable Debug Mode. This enables the M-Port Debug menu, in which you can go directly to the isolated storage location and execute queries to a Mamut database.

To enable Debug Mode, browse to the `HKEY_CURRENT_USER\Software\RedAnt\RedAnt M-Port` key in the *Registry Editor* and create a new DWORD (32-bit) value called *Debug*. The value should be 1. You need to restart M-Port in order to effectuate changes in Debug Mode.

Alternatively, you can press the F6 key when no workflow has been loaded. Note that this doesn't work when you have only one workflow, because that will be automatically selected.

3 Connections

A connector, be it origin or destination, creates a connection to a data location. Such a location may be Mamut, but also a file, a webservice or a database. The Mamut connector, for instance, creates a connection to a particular Mamut company database. If a connector is used as an origin connection, the connection will be made for reading, whereas a destination connection is used for writing (and, where required, reading too).

A connector has a `type`-attribute, one of:

- Mamut
- E-mail
- File
- Folder
- FTP
- MySQL
- SQL (for Microsoft SQL Server)
- Web (for SOAP webservices)

Optional and required attributes for these are specified below. To allow an *end-user*, instead of you as an integrator, to specify a connection property, the string *{user}* may be used to denote such fields.

3.1 Email

Can be used only for a destination connection. Sends an e-mail message. In both the to, from, sender, subject, message and filename fields, you can use field names between curly braces, such as

{Person.FirstName} and *{Order.InvoiceID}*.

node name	description
`server`	The SMTP server that is used to send the message.
`user` *(optional)*	The username for login.
`password` *(optional)*	The password to use for the connection.
`from`	The e-mail address the message appears to be sent from.
`sender`	The name of the sender of the message.
`to`	The e-mail address of the recipient of the message. If you specify a field name between curly braces here, your output data will be split for all unique values in this field, and a separate e-mail message will be sent to all of the e-mail addresses so found, with only their respective data.
`subject`	The e-mail message's subject.
`message`	The text inside the e-mail message.
`filename`	The file name of the attachment. May contain *{timestamp}*, which will be replaced by the current date and time in the format *yy-MM-dd-HHmmss*, or (a) field code(s) that are replaced by the actual data, such as *{Order.OrderID}*.
`split` *(optional)*	Contains the name of a field on which to split output data. If specified, will export multiple e-mails, if a field is included in the specified to address.

3.2 File

Read or write a file. All nodes are optional; when no path or filename are given, the user will need to select it from the M-Port interface.

node name	description
`path`	The path in which a file is created.
`filename`	The name of the file that is created. If used in a destination connection, may contain *{timestamp}*, which will be replaced by the current date and time in the format *yy-MM-dd-HHmmss*, or (a) field code(s) that are replaced by the actual data, such as *{Order.OrderID}*.
`encoding`	Determines the encoding of the file, one of ANSI, ASCII, UTF8, UTF16 (Unicode), or Windows1252. If none specified, UTF8 is assumed.
`delete_original_file`	When importing, if this node contains *true*, removes the original file (a backup is stored in M-Port's backup folder). This setting is ignored when the workflow is run as a scheduled task.
`split`	Contains the name of a field on which to split output data. If specified, will export multiple files if a field code is included inside the specified filename.

3.3 Folder

Read multiple files from a folder.

node name	description
`path`	The path from which files should be read.
`filemask`	Determines which files are read, for example **.xml* reads all XML files from the

	given path.
`encoding` *(optional)*	Determines the encoding of the files, one of ANSI, ASCII, UTF8, UTF16 (Unicode), or Windows1252. If none specified, UTF8 is assumed.
`delete_original_files` *(optional)*	If this node contains *true*, removes the original files (a backup is stored in M-Port's backup folder). This setting is ignored when the workflow is run as a scheduled task.

3.4 FTP

Reads or write to and from an FTP server.

node name	description
`server`	The server name or IP address.
`user`	The username for login.
`password`	The password to use for the connection.
`path` *(optional)*	Path to read or store the file from. Prefix the path with a / to specify the full path from the root, or without the / to give a relative path from the initial location.
`filename`	The name of the file that is created. If used in a destination connection, may contain *{timestamp}*, which will be replaced by the current date and time in the format *yy-MM-dd-HHmmss*, or (a) field code(s) that are replaced by the actual data, such as *{Order.OrderID}*.
`mode`*(optional)*	Should contain either *append* or *overwrite*. If not specified, *overwrite* is assumed.
`enablessl`*(optional)*	If this node contains *true*, a secure SSL connection to the server is attempted.
`encoding` *(optional)*	Determines the encoding of the file, one of ANSI, ASCII, UTF8, UTF16 (Unicode), or Windows1252. If none specified, UTF8 is assumed.
`delete_original_file` *(optional)*	When importing, if this node contains *true*, removes the original file (a backup is stored in M-Port's backup folder).
`split` *(optional)*	Contains the name of a field on which to split output data. If specified, will export multiple files if a field is included in the specified filename.

3.5 Mamut

Read or write a Mamut-database. In order to use a Mamut Connector, Mamut needs to be installed on the same machine as M-Port. All nodes are optional; when no instance or (system) database is given, the user will need to select those from the M-Port interface.

node name	description
`instance`	The name of the server and instance that Mamut is located on, for example, *SERVER/MAMUT*. If you specify *{user}* here, the user will be presented with a SQL server network browser to select a database instance. In addition, the contents of the `database`-node will be ignored and *{user}* will be assumed.
`systemdatabase`	The number of the system database to connect to. If you specify *{user}* here, the user will be presented with a list of system databases to pick from. If no system database number is specified, 1 will be assumed.
`database`	The number of the company database to connect to. If you specify *{user}* here, the user will be presented with a list of company databases to pick from.

3.6 MySQL

Read from or write to a MySQL-database. In order to use the MySQL connector, you will need to install MySQL Connector/Net 6.1.3, from http://dev.mysql.com/downloads/connector/net/6.1.html. The MySQL connector is probably best used using the Query format.

node name	description
`server`	The name or IP address of the server you want to connect to, for example, *mysql.company.org*.
`database`	The name of the database to connect to.
`user`	User name.
`password`	Password.

3.7 SQL

Read from or write to a Microsoft SQL Server-database. The SQL connector is probably best used using the Query format.

node name	description
`instance`	The name of the server and instance you want to connect to, for example, *SERVER/COMPANY*.
`database`	The name of the database to connect to.

3.8 Web

Read from or write to a SOAP webservice.

node name	description
`server`	The server name or IP address (leave out the 'http' prefix).
`enable_ssl` *(optional)*	To connect over HTTPS instead of HTTP, put 'true' in this field.
`request` *(optional)*	When retrieving data from a webservice, this field contains the XML that is posted to retrieve information.
`action` *(optional)*	The SOAP action.
`expected_response` *(optional)*	The response that is returned when posting to a webservice, is checked to contain this text.
`split` *(optional)*	Contains the name of a field on which to split output data. If specified, will do multiple posts to the webservice.

4 Formats

In general terms, a format is a data definition. It is a kind of dictionary, that translates incoming data to the M-Port intermediate format, or the intermediate format to outgoing data.

Below, all settings for a node will be specified. An input or output format at least contains the `type` attributes, which is one of:

- CSV:

- CSV;
- Fixed;
- Mamut;
- Query (for performing database commands);
- Report (for exporting to HTML, PDF, Microsoft Excel and Microsoft Word through Crystal Reports);
- XML;
- Word.

4.1 CSV

Comma, semicolon, of differently separated format.

node name	description
`template`	Specifies the name of the CSV file you'll be using as a document template. Do not insert the .csv extension, as this will be added automatically.
`delimiter` *(optional)*	The delimiter of the CSV file, for example `<delimiter>,</delimiter`. If no delimiter is specified, **;** will be assumed.

Template

```
Product identification;Description;Can be ordered;Price
{Product.ProductID};{Product.Description};YES of course!;{Product.Price}
```

This template allows you to read and write CSV files that follow this specification. If two rows are specified inside the CSV, we assume the first is the header row and the second contains M-Port field codes. If the CSV template contains only one row, we assume the CSV file contains no header row. If a header row is specified and the CSV format is used for importing, the first row of the CSV file is ignored. If it is specified and the CSV format is used for exporting, it will be outputted as the first row in the CSV file.

4.2 Mamut

node name	description
`action` *(optional)*	Only applies to output formats. If this node contains the value *insert*, only new data will be inserted into Mamut, leaving existing data unmodified. If *replace* is specified, or if the node is not present, items are created if they do not exist, or updated if they do. Finally, if *update* is specified, only existing records will be updated and no new entries will be created.
`invert_negatives` *(optional)*	If this node contains the value *true*, negative amounts and the corresponding prices will be inverted. Please see below for an explanation.
`force_columns` *(optional)*	When used as an origin format, forces only the supplied columns (separated by a space) to be used for input.
Query modificators *(optional)*	You can modify the M-Port generated query using the nodes `select`, `limit`, `where`, `order_by` and `join`. Note that knowledge of the Mamut database internals is required to properly use this functionality.

All required information for reading or writing this data is built in M-Port. Hence, all relevant data as described in the M-Port intermediate format, can be written to and from Mamut, unless noted otherwise.

Example

- <where>[invoice].[standard] = 1 AND [registered].[standard] = 1 AND [g_cpers].[inactiv] =

0</where> (Voor alleen het standaard adres en actieve contactpersonen)

4.3 Query

You can read data from a database, or write to it, using the Query format. You can manually enter the SQL query that is to be used, directly in the workflow.

node name	description
`query`	Specifies the SQL query.
`start_statement` *(optional)*	Specifies a SQL statement that will be executed once, at the beginning of the session, before any other query will be done (except when testing workflows).
`end_statement` *(optional)*	Specifies a SQL statement that will be executed once, at the end of the session, after all other queries have been done (except when testing workflows).

When reading, you need to use M-Port field syntax (will be rewritten to discard {} and rewrite dots to underscores). For example:

```
SELECT
id AS {ORDER.OrderID},
reference AS {ORDER.Reference}
FROM TABLE
```

This will retrieve the data in the columns *id* and *reference* into M-Port's data model, in the Order table, fields ID and Reference.

When writing data, you can insert fieldcodes in the SQL query, which will be substituted by M-Port when executing the query.

```
INSERT INTO invoiceTable SET reference = "{order.Reference}"
WHERE id = {ORDER.InvoiceID}
```

4.4 Report

You can export data to PDF, HTML, Microsoft Excel or Microsoft Word using Crystal Reports. The Crystal Reports modules come bundled with M-Port, but M-Port, as nearly all Crystal Reports-enhanced applications, does not include a report designer. Create a Crystal Reports in your designer using M-Port's Model as the data source. You can then save your report as a .rpt file. This file is used as the template in M-Port.

node name	description
`template`	Specifies the name of the Crystal Report you'll be using as a document template. Do not insert the .rpt extension, as this will be added automatically.
`type`	As a report type, you can specify one of *doc, html, pdf, xls*, for exporting to Word, HTML, PDF, or Excel respectively.

4.5 Word

You can export your data to a Microsoft Word document only if you have Word installed on the computer M-Port runs on. Create a Word template by saving your Word document as a document template (.dot) and inserting merge fields. In Word 2003, these can be inserted using *Insert > Field...*, Select *Merge field* from

the list, and type a field code in the *Field name* field. Word uses the format *Datatype_Field* for M-Port merge fields instead of *Datatype.Field* as is customary in other templates.

If you want to iterate through all the rows in your data, create a Word table and make sure the last field in the row you want to iterate is a {NEXT} field (Select *Next* from the *Field names* list, don't type this as a merge field name!).

If your M-Port has debug mode enabled, you can export a CSV file with all possible field codes. You can use this CSV file as a data source in Word, so you can select all possible fields from a drop-down list instead of typing them in.

node name	description
`template`	Specifies the name of the Word template you'll be using as a document template. Do not insert the .dot extension, as this will be added automatically.
`split` *(optional)*	Contains the name of a column on which to split output data. If specified, will create and export to multiple output connectors.

4.6 XML

Extensible Markup Language format, that is easily readable for both machines and humans alike.

node name	description
`template`	Specifies the name of the XML file you'll be using as a document template. Do not insert the .xml extension, as this will be added automatically.
`write_short_tags` *(optional)*	If this node contains the value *false*, abbreviated empty XML tags (e.g., `<short />`) will be written as regular empty tags (`<short></short>`).
`write_empty_tags` *(optional)*	If this node contains the value *false*, empty XML tags (e.g., `<short></short>`) will be stripped from the output. This won't happen if the tag contains attributes!

Template

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<orders>
        <order id="{Order.OrderID}">
                <orderline id="{Order.LineNumber}">
                        <product id="{Order.LineProductID}" price="{Order.LinePrice}">
                                <description>{Order.LineProductDescription}</description>
                        </product>
                </orderline>
        </order>
</orders>
```

Thus, an XML definition file is a valid XML-file, in which the value of attributes or textnodes contain an M-Port field code. Please note that M-Port currently is not capable of handling XML files that contain duplicate nodes on one level, for example, a `<Reference>`-node that contains two `<Data>`-nodes that cannot be distinguished. In this case, you will get duplicate data rows for every duplicate occurrence.

Merge nodes

M-Port tries to fit generated lines into the output file as neatly as possible. In the example above, *order* is used as the input. So, each line of the order contains the order header fields, such as *{Order.OrderID}*. M-

Port tries to put data that belongs to the same order ID under the respective node. The effect is that output won't be like *this*:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<orders>
        <order id="1">
                <orderline id="1.1">
                        <product id="STRAWBERRY" price="1.95" />
                </orderline>
        </order>
        <order id="1">
                <orderline id="1.2">
                        <product id="BANANA" price="2.95" />
                </orderline>
        </order>
        <order id="2">
                <orderline id="2.1">
                        <product id="LEMON" price="3.95" />
                </orderline>
        </order>
</orders>
```

But rather will be formatted like *this*:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<orders>
        <order id="1">
                <orderline id="1.1">
                        <product id="STRAWBERRY" price="1.95" />
                </orderline>
                <orderline id="1.2">
                        <product id="BANANA" price="2.95" />
                </orderline>
        </order>
        <order id="2">
                <orderline id="2.1">
                        <product id="LEMON" price="3.95" />
                </orderline>
        </order>
</orders>
```

M-Port utilizes attributes and text nodes that are direct children of the node that is to be matched.

Simulating unique ID's

Some XML definitions do not contain the unique attribute of the datatype (for example, for orders, that would be *{Order.OrderID}*). That is inconvenient, because it prevents M-Port from properly merging the order nodes.

To solve this problem, you can add a simulation node: `<mport_unique_id>`. This node has to be inserted in the XML template, but will not be exported to the output.

5 Field Codes

5.1 Activity

All the field codes below can be used in the format *{Activity.**xxx**}*. You can access information regarding the

All the field codes below can be used in the format *{Activity.**xxx**}*. You can access information regarding the first associated contact or contact person using *{Contact.**xxx**}* and *{Person.**xxx**}*. In addition, you can access information of the responsible employee using *{Employee.**xxx**}*.

code	description
	activity subject
	user ID of the creator of the activity *(export only)*
	full name of the creator of the activity *(export only)*
	employee ID of the responsible employee
	customer ID of the associated contact
	activity registration date + time
	activity start date + time
	activity end date + time
	activity type *(will **not** be automatically created if new)*
	activity type *(number from standard register)*
	status *(when importing, will be created if new)*
	status ID *(number from standard register)*
	project
	project ID
	activity priority: 1 = None, 2 = High, 3 = Medium, 4 = Low
	finished *(boolean; use true or false)*
	activity notes

5.2 Contact

All the field codes below can be used in the format *{Contact.**xxx**}*. You can access information regarding the default contact person using *{Person.**xxx**}*.

code	description
	customer name
	customer ID
	supplier ID *(exporting only, use IsSupplier = true to generate a supplier number)*
	customer *(boolean; use true or false)*
	supplier *(boolean; use true or false)*
	partner *(boolean; use true or false)*
	private *(boolean; use true or false)*
,	phone numbers
,	fax numbers
	e-mail address
	www
	language *(when importing, will be created if new)*
	language ID *(number from standard register)*
	organisation number (i.e., KvK-nummer)
	department *(when importing, will be created if new)*
	department ID *(number from standard register)*
	List of comma-separated groups (i.e., *Group A*, *Group B*). When importing, make sure these groups

already exist in the standard register. Existing groups for the contact will be removed when you import groups.

street name and number

zip code

city

region

country name *(use only for exporting)*

two-letter country ISO code

whether or not the contact is a main office *(export only; boolean -- use true or false)*

name of the main office if this contact is a branch office *(export only)*

customer ID of the main office, if this contact is a branch office

status *(when importing, will be created if new)*

status *(number from standard register; preferred for importing)*

line of business *(when importing, will be created if new)*

line of business *(number from standard register; preferred for importing)*

category *(when importing, will be created if new)*

category *(number from standard register; preferred for importing)*

response type *(when importing, will be created if new)*

response type ID *(number from standard register; preferred for importing)*

username (i.e., *demo*) of an employee *(see User administration in Mamut)*

user id of an employee *(export only)*

project

project ID *(number; preferred for importing)*

name of the contact's primary bank

number of the contact's primary bank account

owner of the primary bank account

primary bank account type *(**1** - Post, **2** - Bank)*

currency *(three letter ISO code)*

currency ID *(number from standard register)*

type of reminder (use for direct debit, ***0*** *- (none),* ***10*** *- Payment Reminder,* ***31*** *- Factoring,* ***41*** *- Direct debit*)

VAT number

discount percentage

credit limit (for a client)

whether or not the account is on hold *(export only; boolean - true or false)*

payment terms

payment terms *(number from standard register; preferred for importing)*

delivery method *(export only)*

delivery method ID *(number from standard register, w_delitypes) (export only)*

delivery terms *(export only)*

delivery terms ID *(number from standard register) (export only)*

VAT code for sales
*(**2** - BTW plichtig, **3** - Vrijgestelde klant, **4** - Export buiten EU, **5** - Export EU, **6** - Export naar EU-derden)*

VAT code for purchasing

	*(**1** - Vrijgesteld, **2** - BTW plichtig, **7** - Import binnen EU, **8** - Import buiten EU, **9** - EU driehoeks import)*
	contact memo

5.3 Currency

All the field codes below can be used in the format *{Currency.**xxx**}*.

code	description
	currency symbol (i.e., *NOK*)
	divider (usually 1)
	exchange rate for foreign currency (i.e., 1 NOK ≈ *0.12* EUR)

5.4 Employee

All the field codes below can be used in the format *{Employee.**xxx**}*.

code	description
	employee id *(required when importing)*
	alias or user name ("employee id 2")
	full name (first + middle + surname) *(export only)*
	first name
	middle name
	surname
	Initials
	department *(when importing, will be created if new)*
	department ID *(number from standard register)*
	gender *(numeric; **1** - male, **2** - female)*
	(home) phone number
	cell phone number
	e-mail address

5.5 Journal Transactions

All the field codes below can be used in the format *{Journal.**xxx**}*. When exporting, you can access contact properties of journals using *{Contact.**xxx**}*, and the associated project information using *{Project.**xxx**}*. Imported journals will show up in Mamut at the *Journal Entry* page (*Inboeken* in Dutch). If you click the settings button, you can choose to *Show journals for all users*. This will show all imported transactions.

When importing journal transactions, it is required to give all lines of one transaction the same *Key*. This should be negative in order to allow Mamut to generate its own ID, but still be equal for all related lines. If not, Mamut will give an API error that the journal is not balanced.

	code	description
Header		journal ID
		registration date

	period
	year
	day book *(letter code as displayed in Mamut, for instance **B** = bank)*
	day book ID *(number from standard register)*
Lines	entry number *(export only)*
	account number *(be sure to include leading zeroes)*
	description
	currency code (i.e., EUR)
	currency ID from g_currency
	exchange rate for this currency
	saldo
	due date
	vat code
	customer ID
	invoice ID
	department
	department ID
	cost layout *(export only)*
	cost center group *(export only)*
	cost center *(export only)*

5.6 Order

All the field codes below can be used in the format *{Order.**xxx**}*. When exporting, you can access product properties of orderlines using *{Product.**xxx**}*, the contact information using *{Contact.**xxx**}*, and the default contact person's information using *{Person.**xxx**}*. When importing, you can check/create new customers using *{Contact.**xxx**}*.

When importing orders, it is required to give all lines of one order the same *OrderID*. This should be negative in order to allow Mamut to generate its own incremental order number, but still be equal for all order lines.

code	description
Header	order ID
	invoice ID
	associated invoice ID (i.e., the original invoice ID for a credit note) *(export only)*
	associated customer's ID
	order status (see g_orderstatus) *(export only)*
	use gross prices (including VAT) for the order lines *(import only)*
	multiline textfield containing the invoice address entered in the Mamut interface
	multiline textfield containing the invoice address entered in the Mamut interface
	order date *(export only)*
	invoice date
	production date *(export only)*
	due date *(export only)*

	delivery date
	our reference *(export only: lastname from employee register)*
	employee ID of our reference
	your reference
	order reference
	project
	project id *(number from standard register)*
	department *(when importing, will be created if new)*
	department id *(number from standard register)*
	order memo text
	pick list text
	delivery note text
	note in 'order information' *(export only)*
	sales status *(export only)*
	net total *(export only)*
	gross total *(export only)*
	VAT total *(export only)*
	discount total *(export only)*
	open balance for this invoice *(export only)*
	payment terms
	payment terms ID *(number from standard register)*
	direct debit (automatische incasso) *(boolean; use true or false)*
	delivery method
	delivery method ID *(number from standard register, w_delitypes)*
	delivery terms *(export only)*
	delivery terms ID *(number from standard register) (export only)*
	response type
	response type ID *(number from standard register)*
	freight bill text
	service order text (left pane)
	service order text (right pane)
	number of items *(export only)*
	number of pallets *(export only)*
	total weight, including order lines and additional order weight *(export only)*
	total volume, including order lines and additional order volume *(export only)*
	shipment number
	whether or not the order is ready for invoicing *(boolean; use true or false)*
	whether or not the order is picked *(boolean; use true or false) (export only)*
	whether or not a credit note has been generated using the X button *(boolean; use true of false - export only)*
	number of order lines in this order *(export only)*
Lines	number of this order line *(export only)*
	type of this orderline *(export only; 0 = text, 1 = product, 2 = product bundle, 3 = part of product bundle)*

	order line product ID
	order line product description
	order line text
	order line product price
	order line quantity ordered
	order line quantity to deliver
	order line quantity delivered *(export only)*
	order line discount price *(export only)*
	order line discount (%)
	order line project
	order line project id *(number from standard register)*
	order line department
	order line department id *(number from standard register)*
	order line tracing number
	order line delivery date
	order line net price *(export only)*
	order line gross price *(export only)*
	order line VAT amount *(export only)*
	order line VAT rate *(export only)*
	order line VAT code *(number from standard register)*
	order line warehouse

5.7 Person

All the field codes below can be used in the format *{Person.**xxx**}*. When exporting, you can address the associated contact information the person belongs to using *{Contact.**xxx**}*.

code	**description**
	Unique contact person ID *(required when, during imports, updating existing contacts in Mamut)*
	customer id to which this person belongs *(required when importing)*
	full name (first + middle + surname) of the person *(export only)*
	first name
	middle name
	surname
	initials
	salutation
,	titles
,	phone numbers
,	fax numbers
,	cell phone numbers
,	e-mail addresses
	homepage
	department *(when importing, will be created if new)*
	department ID *(number from standard register)*
	project *(when importing, will be created if new)*

	project ID *(number from standard register)*
	List of comma-separated groups (i.e., *Group A, Group B*); *(export only)*
	job title *(when importing, will be created if new)*
	job title ID *(number from standard register)*
	birthdate
	rank *(when importing, will be created if new)*
	rank ID *(number from standard register)*
	ID, such as the social security number or the Dutch BSN
	Active or incative *(Export only)*
	Default contact person or not *(Export only)*
	street name and number
	zip code
	city
	region
	two-letter country ISO code
	memo field

5.8 Product

All the field codes below can be used in the format *{Product.**xxx**}*.

	code description
Codes	product ID
	product name / description
	EAN code
	commodity code
	the industry's product no.
	EU Intrastat code
	parent product ID *(export only)*
Variants	whether or not this product is a main product for variants *(export only; boolean; use true or false)*
	whether or not this product is a variant of another product *(export only; boolean; use true or false)*
Product Bundles	whether or not this product is composed of other products *(export only; boolean; use true or false)*
	whether or not this product is part of a structure *(export only; boolean; use true or false)*
	structure this component belongs to, when inside a structure *(Sales price and purchase price for the parent product will not be automatically updated)*
	quantity the article occurs inside a structure
Properties	weight
	weight unit *(when importing, will be created if new)*
	weight unit *(number from standard register)*
	unit description *(when importing, will be created if new)*
	unit *(number from standard register)*
	quantity

	volume
	volume unit *(when importing, will be created if new)*
	volume unit *(number from standard register)*
	delivery time (in days)
	product groups *(when importing, will be created if new)*
	product group ID's *(numbers from standard register)*
	product category *(when importing, will be created if new)*
	product category ID *(number from standard register)*
	project *(when importing, will be created if new)*
	project ID *(number from standard register)*
	department *(when importing, will be created if new)*
	department ID *(number from standard register)*
Settings	product is a stock item *(boolean; use true or false)*
	product is work/service *(boolean; use true or false)*
	product is a resource *(boolean; use true or false)*
	product is a campaign product *(boolean; use true or false)*
	required serial number for receiving goods and withdrawing goods *(boolean; use true or false)*
	required consignment number for receiving goods and withdrawing goods *(boolean; use true or false)*
	required best before date for receiving goods and withdrawing goods *(boolean; use true or false)*
Price	price
	purchase price
	costs
	cost price *(export only)*
	recommended sales price excl. VAT
	override cost price in sales module *(boolean; true or false)*
Individual	discount percentage for an individual customer *(export only; requires one or more contact-fields)*
	price for an individual customer, not taking into account the discount percentage *(export only; requires one or more contact-fields)*
	price for an individual customer *(export only; requires one or more contact-fields)*
Surcharges	surcharge ID *(number from standard register)*
	surcharge group text *(when importing, make sure this text is unique in Mamut)*
	surcharge description text *(when importing, make sure this text is unique in Mamut)*
VAT	VAT ID for sales *(number from standard register)*
	VAT rate for sales *(export only)*
	VAT ID for purchases *(number from standard register)*
	VAT rate for purchases *(export only)*
Accounts	Balance sheet account for stock value
	Account for stock change
	Cost account for product cost
	Sales account liable to VAT
	VAT exempt sales N/C

	VAT exempt sales N/C
	Sales account export outside EU
	Sales account export to EU
Notes	product information
	sales information
	technical information
	support information
	production information
	other information
	automatically transfer product information to product lines in sales order *(boolean; use true or false)*
Supplier	default supplier's name
	ID of default supplier
	product ID of default supplier
	EAN code of default supplier
	Delivery time of default supplier
	default supplier's product description
	use default supplier's product description on orders *(boolean; use true or false)*
	purch. price in default supplier's currency
	default supplier's recommended sales price
	discount percentage on purchase
Stock	default warehouse id *(number from standard register; when importing, only one warehouse can be imported per product in one run)*
	location inside warehouse *(only for standard warehouse and standard location, M-Port can only import one location)*
	ID from location inside warehouse *(only for standard warehouse and standard location, M-Port can only import one location)*
	stock in default warehouse *(import: integer numbers only)*
	combined stock in all warehouses *(export only)*
	combined available stock (stock - to customers + from vendors) in all warehouses *(export only)*
	minimal stock in default warehouse *(import: integer numbers only)*
	maximum stock in default warehouse *(import: integer numbers only)*
	next delivery date (based on outstanding purchase orders) *(export only)*
	stock order level in default warehouse *(import: integer numbers only)*
	stock order quantity for default warehouse *(import: integer numbers only)*
Webshop	show product on webshops *(boolean; use true or false)*
	brief default description on webshops, may contain HTML
	detailed default description on webshops, may contain HTML
Misc	link to picture (path)
	link to alternative picture (path)
	product website
	whether the product is an inactive product *(boolean; use true or false)*. Only active products are exported, unless this field is part of the output format, in which case also inactive products are exported.
	if the product is inactive, this product is used as a replacement *(export only)*

5.9 Project

All the field codes below can be used in the format *{Project.**xxx**}*. When importing, either supply an existing project ID or existing project name, and this project will be updated. Otherwise, it will be created.

code	description
Identifiers	project ID
	name
	class
	description
Properties	name of the responsible employee for this project *(export only)*
	employee number of the responsible for this project *(export only)*
	project status *(when importing, will be created if new)*
	project status ID *(preferred for importing; number from standard register)*
	project completion *(when importing, will be created if new)*
	project completion ID *(preferred for importing, number from standard register)*
	project note
Dates	planned start date
	planned end date
	start date
	end date
Budget	number of hours registered
	number of invoiceable hours
	number of paid hours
	total transferrable to sales and invoicing
	total transferrable to payroll
	total invoiced
	total stock invoiced
	total services invoiced
	total invoiced other
	total purchases
	operational revenue
	operating expenses

5.10 Purchase order

All the field codes below can be used in the format *{PurchaseOrder.**xxx**}*. When exporting, you can access product properties of orderlines using *{Product.**xxx**}*, the contact information using *{Contact.**xxx**}*, and the default contact person's information using *{Person.**xxx**}*. When importing, you can check/create new customers using *{Contact.**xxx**}*.

code	description
Header	primary key of invoice order
	primary key of contact
	id of purchase order

	id of purchase order
	status of purchase order
	id of customer
	id of vendor
	invoice address
	delivery address
	register date *(export only)*
	order date *(export only)*
	date of delivery
	date of last change *(export only)*
	your reference
	net total *(export only)*
	gross total *(export only)*
	vat total *(export only)*
	discount total
Lines	line id
	product's primary key
	id of product

5.11 Timesheet

All the field codes below can be used in the format *{Timesheet.**xxx**}*. When exporting, you can access contact information of the contact associated with a timesheet line using *{Contact.**xxx**}*.

	code	**description**
Header		timesheet id *(required when importing, use -1, -2, -3 etc to separate multiple timesheets in one import)*
		timesheet description
Lines		employee ID *(numeric)*
		timesheet date
		start time on this timesheet line
		end time on this timesheet line
		number of hours worked
		customer ID
		project ID *(number from standard register)*
		department ID *(number from standard register)*
		product number
		time line description
		time type *(number from standard register)*
		time type
		transfer timesheet line to project
		transfer timesheet line to order
		transfer timesheet line to payroll

6 Processors

A processor processes input data, before forwarding data to the output-format. A workflow-XML file is necessary to specify configuration settings for a given processor.

6.1 AddLines

This function can be used to add an extra line to groups of rows.

Configuration

- `key_column`: field that is used to identify which rows are grouped together
- `sum_fields` *(optional)*: fields in the row whose values need to be added. If not specified, all data columns that contain numeric data are summed.
- `no_summing` *(optional)*: when true, do not sum any data column *(boolean; use true or false)*
- `where` *(optional)*: specifies which rows are processed for summing
- `column` *(optional)*: in this column, a value is stored based on the specified template, so added rows can be differentiated from original data
- `template` *(optional)*: works like the Merge-processor's template, may contain M-Port field codes and is required when *column* is specified
- `force_double` *(optional)*: treat all values in sum_fields as double (floating point) fields *(boolean; use true or false)*

Example

```
<processor type="AddLines">
  <key_column>order.OrderID</key_column>
  <sum_fields>order.LineQuantity product.Weight</sum_fields>
  <column>temp.IsAdded</column>
  <template>1</template>
</processor>
```

6.2 AddressSplit

Allows a user to split a Mamut order address text field, composing of multiple lines, into separate fields. If these fields already contain an address which perfectly matches the address in the address text field, no dialog will be presented to the user.

Configuration

- `id_column`: specifies the ID on which to split orderlines, usually Order.OrderID or Order.InvoiceID
- `address_column`: the multiline address text field of the order, for instance Order.DeliveryAddress or Order.InvoiceAddress
- `street_column`, `zip_column`, `city_column`, `countryiso_column`: the separate address fields. The country ISO column contains the two-letter ISO code that's used within Mamut, e.g. Contact.DeliveryCountryISO.

Example (uses a temp table)

```
<processor type="AddressSplit">
  <id_column>order.orderid</id_column>
  <address_column>order.deliveryaddress</address_column>
  <street_column>temp.deliverystreet</street_column>
  <zip_column>temp.deliveryzip</zip_column>
  <city_column>temp.deliverycity</city_column>
  <countryiso_column>temp.deliverycountryiso</countryiso_column>
</processor>
```

Note: when working on orders, keep in mind to use a `temp` table for the address fields! If you use the Contact.*-fields, you will change the address of all orders of that customer for this workflow execution to the *first* split address.

6.3 Concat

Adds data to a specified column.

Caution If the given data in a specified column is not of type string the column is converted to string. The consequence of this action is that numerical Select-processors can't be applied, e.g.: (`invoiceid` > 10) is impossible when the column is converted!

Configuration-nodes

- `column`: the column that has to be edited
- `prepend`: prepending text *(optional)*
- `append`: appending text *(optional)*

Example

```
<processor type="Concat">
  <column>product.productid</column>
  <prepend>1234</prepend>
  <append>000000</append>
</processor>
```

The result of this action would be something like the following: `1234DATA000000`.

6.4 KeyValue

Apply if/then/else business logic using queried key/value pairs.

Configuration

- `query`: query to retrieve key/value pair table from the database
- `key_column`: the column that keys correspond to
- `value_column`: the column the values correspond to and results will be written to
- `if`: logical expression that will be evaluated, determining whether the results from 'then' or 'else' will be used as result
- `then`: expression that will be evaluated if the 'if' test succeeds
- `else`: expression that will be evaluated if the 'if' test fails
- `calculation`: will determine the new value in key/value pair table after results have been evaluated (use *{value}* for the initial value or *{result}* for the value from 'then' or 'else')

Example: Used for applying readily available stock as 'to be delivered' in open orders

```
  <processor type="KeyValue">
    <query><![CDATA[
      SELECT g_prod.PRODID,
      ISNULL((SELECT SUM([g_storelink].[stock] - [g_storelink].[tocustomers]) FROM [g_sto
      GROUP BY [g_storelink].[fk_product]), 0)
      FROM g_prod
    ]]></query>
    <key_column>order.LineProductID</key_column>
    <value_column>order.LineQuantityToDeliver</value_column>
    <if><![CDATA[{value} >= {order.LineQuantityToDeliver}]]></if>
    <then><![CDATA[{order.LineQuantityToDeliver}]]></then>
    <else><![CDATA[0]]></else>
    <calculation><![CDATA[{value} - {result}]]></calculation>
  </processor>
```

First, this retrieves products and available stock as key/values from the database, using the *query*. This gives a data table like this:

```
ART1   15
ART2   20
ART3   -3
```

Example: Used for applying readily available stock as 'to be delivered' in open orders and works with negative stock

```
  <processor type="KeyValue">
    <query><![CDATA[
      SELECT g_prod.PRODID,
      ISNULL((SELECT SUM([g_storelink].[stock] - [g_storelink].[tocustomers]) FROM

[g_storelink] WHERE [g_storelink].[fk_product] = [g_prod].[pk_prodid]
      GROUP BY [g_storelink].[fk_product]), 0)
      FROM g_prod
    ]]></query>
    <key_column>order.LineProductID</key_column>
    <value_column>order.LineQuantityToDeliver</value_column>
    <if><![CDATA[Math.Max(0, {value}) >= {order.LineQuantityToDeliver}]]></if>
    <then><![CDATA[{order.LineQuantityToDeliver}]]></then>
    <else><![CDATA[Math.Max(0, {value})]]></else>
    <calculation><![CDATA[{value} - {result}]]></calculation>
  </processor>
```

For each row in the data, the following steps are then taken:

- The *key_column* (i.e., the product ID) is used to lookup a value (i.e., the available stock) in this table.
- If the *if*-expression succeeds, the *value_column* (the quantity to deliver) is set to the result of the *then*-expression. If it fails, it's set to the result of the *else*-expression, in this case, 0.
- Finally, the value in the key/value data table is changed using the result of the *calculation*-expression.

In this case, let's look at order lines that contain ART2 (currently, 20 in stock). Suppose the first order line contains 30 ART2's. The available stock is not enough to satisfy this order (*if*: 20 >= 30, fails), so its quantity to deliver will be set to 0 (*else*) and the new value in the key/value data table is set to 20 - 0 (*calculation*). Now suppose the second order line contains 15 ART2's. In that case, *if* succeeds (20 >= 15), so the quantity to deliver is set to 15 (*then*) and the new value in the key/value data table will be 5 (20 - 15). In that case, if the third order line contains more than **5** items, the quantity to deliver will be set to 0, et cetera.

6.5 LookUp

Looks up values in the Mamut database, based on an SQL query supplied in the processor.

Configuration

- `to_column`: the column in which the result of the query will be placed
- `query`: an SQL query on the Mamut database, containing field names enclosed in curly braces, e.g. {Order.InvoiceID}
- `default_value` *(optional)*: specifies a default value in case the query did not return a result. You can use field names for the default value as well, such as {Order.OrderID}.

Example (looks up order ID based on reference)

```
<processor type="LookUp">
  <to_column>order.orderid</to_column>
  <query>SELECT orderid FROM g_order WHERE reference = '{order.reference}'</query>
  <default_value>0</default_value>
</processor>
```

Example 2 (looks up contact person ID based on first name, last name and associated customer ID)

```
<processor type="LookUp">
  <to_column>person.Key</to_column>
  <query><![CDATA[
    SELECT cpersid
    FROM g_cpers
    LEFT JOIN g_contac ON g_contac.contid = g_cpers.contid
    WHERE firstname LIKE '{person.FirstName}' AND lastname LIKE '{person.Lastname}' AND cu
 ]]></query>
 <default_value>{person.Key}</default_value>
</processor>
```

Example 3 (looks up the value of the custom field 'Branchecode', based on a customer ID)

```
<processor type="LookUp">
  <to_column>temp.Branchecode</to_column>
  <query><![CDATA[
    SELECT fieldvalue
    FROM g_cfield
    LEFT JOIN g_clisys ON g_cfield.fieldname = g_clisys.nr AND g_clisys.id = 133
    LEFT JOIN g_contac ON g_contac.contid = g_cfield.contid
    WHERE g_cfield.cpersid IS NULL
    AND g_clisys.descr LIKE 'Branchecode'
    AND g_contac.custid = {contact.CustomerID}
]]></query>
</processor>
```

Example 4 (looks up a customer ID based on the value in the custom field 'EAN code')

```
<processor type="LookUp">
  <to_column>contact.CustomerID</to_column>
  <query><![CDATA[
    SELECT TOP 1 custid
    FROM g_contac
    LEFT JOIN g_cfield ON g_cfield.contid = g_contac.contid
    LEFT JOIN g_clisys ON g_cfield.fieldname = g_clisys.nr AND g_clisys.id = 133
    WHERE g_cfield.cpersid IS NULL
    AND g_clisys.descr LIKE 'EAN code'
    AND g_cfield.fieldvalue = '{temp.EANCode}'
]]></query>
</processor>
```

6.6 Match

Compares import data to existing records in your database. If a perfect match on a number of columns is found, the data is matched to this particular record. In other cases, you can manually select to either:

- update an existing record (the closest match is automatically suggested by the processor), or
- import the data as a new record, or
- not import the data at all.

Configuration

- `datatype`: specifies the datatype on which this processor operates
- `from_column`: source for values to select as destination
- `match_columns`: column(s) on which to look for perfect matches (these need to exist in both the result of the datatype query and in the input data)
- `match_data_for` *(optional)*: column for which value should be equal to be considered match data
- `to_column`: destination value
- `display_string`: string template that is displayed in the match dialog
- `show_perfect_matches` *(optional)*: if true, handle perfect matches like non-perfect matches
- `ignore_punctuation` *(optional)*: if false, does take into account the characters .,-_':;|? {}#@$%^&* and space

Example

```
<processor type="Match">
  <datatype>contact</datatype>
  <from_column>contact.customerid</from_column>
  <match_columns>contact.name contact.invoicezip</match_columns>
  <to_column>order.customerid</to_column>
  <display_string>{contact.name} ({contact.invoicezip})</display_string>
</processor>
```

This example would try to match contact.Name and contact.InvoiceZip columns that exist in both the input data and the match data. Match data is obtained by querying the database specified by the processor's connector for data of the specified datatype. In this case, the workflow could use order as datatype, whereas this match processor compares only contact data. The result of the query, the contact.CustomerID value, is placed in the order.CustomerID field of the input data.

Example 2

```
<processor type="Match">
  <datatype>person</datatype>
  <match_columns>person.LastName person.FirstName</match_columns>
  <from_column>person.Key</from_column>
  <to_column>person.Key</to_column>
  <match_data_for>person.CustomerID</match_data_for>
  <display_string>{contact.Name}: {person.FirstName} {person.LastName}</display_string>
</processor>
```

This example would try to match contact persons on existing persons -- but only for persons that share the same customer ID as the person that is to be imported.

6.7 Math

Perform basic mathematic or string operations on data.

Configuration

- `column`: name of column that will contain the result of the operation
- `expression`: expression to evaluate, containing string data
- `where` *(optional)*: only performs calculations on specific rows

Example

```
<processor type="Math">
   <column>order.lineproductprice</column>
   <expression><![CDATA[double.Parse("{order.lineproductprice}") / double.Parse("1.19")]]>
</processor>
```

Example 2

```
<processor type="Math">
   <column>order.lineproductprice</column>
   <expression><![CDATA[double.Parse("{order.lineproductprice}") * double.Parse("1.45")]]>
   <where><![CDATA["{order.lineproductid}".StartsWith("EXTRAMARGIN")]]></where>
</processor>
```

Example 3: Combine with a Select processor to remove unwanted data

```
<processor type="Math">
  <column>order.grosstotal</column>
  <expression>0</expression>
  <where><![CDATA["{temp.periodstart}" == "{temp.periodend}"]]></where>
</processor>

<processor type="Select">
  <columns>order.grosstotal</columns>
  <where><![CDATA[`order.grosstotal` > 0]]></where>
</processor>
```

Example 4: See if an invoice has been paid

```
<processor type="Math">
  <column>temp.IsPaid</column>
  <expression><![CDATA[double.Parse("{order.AmountDue}") <= 0.00]]></expression>
</processor>
```

Example 5: Add a remark based on first 4 positions of zip code

```
<processor type="Math">
  <column>order.Reference</column>
  <expression>{order.Reference} - LOCAL</expression>
  <where><![CDATA[StringExtensions.InRange("{temp.Postcode}", "9170, 9172-9179", 4)]]></wh
</processor>
```

Example 6: Put a person's last name if the company name is missing

```
<processor type="Math">
  <column>contact.Name</column>
  <expression>"{person.LastName}"</expression>
  <where><![CDATA["{contact.Name}" == ""]]></where>
</processor>
```

Example 7: Calculate costs based on cost price minus purchase price (because Mamut API can't import cost price)

```
<processor type="Math">
  <column>product.Costs</column>
  <expression>double.Parse("{product.CostPrice}") - double.Parse("{product.PurchasePrice}"
</processor>
```

Example 8: Use supplier description when such a description has been given

```
<processor type="Math">
  <column>product.UseSupplierDescription</column>
  <expression>true</expression>
  <where><![CDATA["{product.SupplierDescription}".Trim().Length > 0]]></where>
</processor>
```

6.8 Merge

This function can be used to merge specified columns to one destination column.

Configuration

- `to_column`: merged results are stored in this column
- `template`: string containing column names between curly braces that will be replaced by the values inside the columns

Example

```
<processor type="Merge">
  <to_column>contact.name</to_column>
  <template>{person.lastname}[, ]{person.firstname}</template>
</processor>
```

Note: if the template contains square brackets [ ], any text between those will be removed when no replacements could be made (i.e., there was no data inside the original columns). In addition, single spaces before or after a replacement field are discarded when the column code is replaced with empty data.

Using fixed values inside the Merge processor's template, you can set values for entire columns:

Example

```
<processor type="Merge">
  <to_column>order.UseLineGrossPrices</to_column>
```

```
  <template>true</template>
</processor>
```
This will set the value inside the UseLineGrossPrices column to true for all lines.

6.9 MergeLines

This function can be used to merge specified rows to one destination row.

Configuration

- `key_column`: field that is used to identify which rows are grouped together
- `sum_fields` *(optional)*: fields in the row whose values need to be added. If not specified, all data columns that contain numeric data are summed.
- `where`: specificies the where-clause the input-data must fulfill
- `product_bundles`: merges all product bundles into the main product *(boolean; use true or false)*

Please note that either `where` or `product_bundles` must be specified, but not both.

Example

```
  <processor type="MergeLines">
    <sum_fields>order.linequantity order.linenetprice</sum_fields>
    <where>{order.lineproductid} like 'shippingcosts'</where>
  </processor>
```

Note: Merging lines that contain `order.LineType` fields, will result in the merged line having the value 1 in this field.

6.10 MultiLookUp

Looks up multiple values in the Mamut database, based on an SQL query supplied in the processor.

Configuration

- `columns`: the columns, separated by spaces, in which the result of the query will be placed
- `query`: an SQL query on the Mamut database, containing field names enclosed in curly braces, e.g. {Order.InvoiceID}

Example (looks up order ID and invoice ID based on reference)

```
<processor type="MultiLookUp">
  <columns>order.orderid order.invoiceid</columns>
  <query>SELECT orderid, invoiceid FROM g_order WHERE reference = '{order.reference}'</que
</processor>
```

6.11 Normalize

Replace non-ASCII (ë) characters with ASCII values (e). When done, it removes any character that's not one of a-z, A-Z, 0-9, a space ( ) or a hyphen (-).

Configuration

- `columns`: names of columns that should be normalized, separated by a space

Example

```
<processor type="Normalize">
   <columns>order.invoiceaddress order.deliveryaddress</columns>
</processor>
```

6.12 OneTime

Lets values in a column through only one time. Any time thereafter, these values will not be exported by this processor. You can use this, for instance, to keep track of order ID's that you have already sent through a warehouse.

Configuration

- `column`: name of column that contains values
- `write_updates` *(optional)*: true or false, determines whether updates are written to the data file (if false, you can use the data file as a set-once exclusion list)

Example

```
<processor type="OneTime">
  <column>order.invoiceid</column>
</processor>
```

Note: a log of passed through values is stored in the `processors/data` folder.

Attention: A OneTime processor should always be *last* in the list of processors, since it takes into account only changes that have been made *before* it is processed. Therefore, it does not play well with the UserSelect processor since, ideally, you would want to discard already-exported data from the presented list. Alas, this is currently not possible.

6.13 RegExSplit

This function can be used to split data over multiple columns, using regular expressions.

Configuration

- `column`: name of column that contains the data that should be processed.
- `regex`: the regular expression that matches against the input.
- `matches`: names of the columns that should be used to list the data matched by the regular expression.

Example

```
<processor type="RegExSplit">
  <column>adrinvoice</column>
  <regex><![CDATA[
      ((?:\s?[\p{L}]+)+)\s?     #streetname
      (\d+(?:\s?\d+)?.*)\r\n    #number
      (.*)\r\n                  #city
      (\d{4}\s?\w{2})\r\n       #zipcode
      (.*)                      #country
    ]]></regex>
  <matches>street number city zipcode country</matches>
</processor>
```

The column *adrinvoice* is used as input to the regular expression. Regex hits are grouped and these results are then placed in the specified columns

are then placed in the specified columns.

Important: It's necessary to escape whitespace using \s because whitespace is ignored by default.

6.14 Select

Selects specific rows from the input data. Please note that there seem to be issues using two select processors after each other; better use AND within the WHERE node.

Configuration

- `where`: specificies the where-clause the input-data must fulfill

Example

```
<processor type="Select">
  <where>{order.invoiceid} = 1</where>
</processor>
```

Example 2

```
<processor type="Select">
  <where>{order.invoiceid} = 1 AND {order.linenumber} > 2</where>
</processor>
```

Example 3 Producten uitfilteren die niet meegenomen mogen worden in een export.

```
  <processor type="Select">
   <where><![CDATA[{order.lineproductid} <> 'TRANSPORT']]></where>
  </processor>
```

6.15 Sort

Sorts the input data based on one or more fields.

Configuration

- `sort_fields`: specifies the fields on which to sort, comma separated and with optionally the sort direction as ASC (for ascending) or DESC (for descending)

Example: Sort contacts Z-A, and persons inside those contacts A-Z

```
<processor type="Sort">
  <sort_fields>contact.Name DESC, person.LastName</sort_fields >
</processor>
```

6.16 StringMap

This function maps values configured in the XML file with those found in the database.

Configuration

- `column`: name of column.
- `mapping`: specifies the searchstring and the replacestring, separated by '->' tokens.

Example

Example

```
<processor type="StringMap">
  <column>paymethod</column>
  <mapping><![CDATA[
    Automatische incasso -> GB
    Op rekening kopen -> BO
    Vooruitbetalen -> IDE
    iDEAL -> IDE
    Creditcard -> EM
  ]]></mapping>
</processor>
```

6.17 Sum

This function computes values and adds them together for rows that share a characteristic.

Configuration

- `destination`: field name to which the computation is written.
- `expression`: value calculation within the row, to add to the values of other rows.
- `group_by`: fields that share the same value in this field will have their expression results summed in the destination field
- `where` *(optional)*: only process rows for which this expression is true
- `remove_summed_lines` *(optional)*: removes rows which have been summed (convenient if you're summing order totals, for instance)

Example: Calculate number of lines per order

```
<processor type="Sum">
  <destination>order.NumberOfLines</destination>
  <expression>1</expression>
  <group_by>order.OrderID</group_by>
</processor>
```

Example 2: Calculate 6% and 19% VAT amount total per order

```
<processor type="Sum">
  <destination>temp.Vat6Amount</destination>
  <expression>{order.LineVatAmount}</expression>
  <group_by>order.OrderID</group_by>
  <where>{order.LineVatRate} == 6</where>
</processor>

<processor type="Sum">
  <destination>temp.Vat19Amount</destination>
  <expression>{order.LineVatAmount}</expression>
  <group_by>order.OrderID</group_by>
  <where>{order.LineVatRate} == 19</where>
</processor>
```

Example 3: Calculate total turnover per country (export orders)

```
<processor type="Sum">
  <destination>temp.TurnoverTotal</destination>
  <expression>{order.NetTotal}</expression>
  <group_by>contact.InvoiceCountry</group_by>
  <remove_summed_lines>true</removed_summed_lines>
</processor>
```

6.18 UserSelectDateRange

Filter data using a date range.

Configuration-nodes

- `column`: the column on which a date range can be applied
- `start_date` *(optional)*: use a predefined value for the start date field (use *{today}* to use today's date)
- `end_date` *(optional)*: use a predefined value for the end date field (use *{today}* to use today's date)

Example

```
<processor type="UserSelectDateRange">
  <column>order.InvoiceDate</column>
</processor>
```

6.19 UserSelect

Allow the user to filter data using a list of values.

Configuration-nodes

- `column`: the underlying value column on which the filter will be applied
- `display_string`: the display formatting (shown to the user) of the values

Example: Select only contacts from a specific country

```
<processor type="UserSelect">
  <column>contact.invoiceCountryISO</column>
  <display_string>Contacts from {contact.invoiceCountry}</display_string>
</processor>
```

6.20 UserVerify

Adds an in-between verification of the export data, that allows the user to acknowledge or cancel the export.

Configuration-nodes

- `columns`: the columns that are shown in the verification dialog

Example

```
<processor type="UserVerify">
  <columns>order.orderid contact.name</columns>
</processor>
```

7 Scheduling (BETA)

Please mind that workflow scheduling is a capability that M-Port has only recently acquired. Scheduling has not been tested as well as other parts of the product. This means that, for now, we strongly recommend using a backup solution on all files that M-Port processes in scheduling. We do not recommend to use M-Port's scheduling capability for vital business processes.

7.1 Command Line Interface (CLI) version

7.1 Command Line Interface (CLI) version

M-Port comes with a command line interface (CLI) version. This version allows for running workflows without loading the M-Port interface. The EXE can be started from the command line with the following parameters:

parameter	description
`/?`	displays help for command line parameters
`/test`	does not run a workflow, but shows the data that will be written on the console
`[workflow]`	Specifies workflow to load. If no path is specified, workflow will be loaded from datastore location.

7.2 Workflow Configuration

In the workflow, insert a `<scheduling>` section, such as this:

```
<scheduling>
  <log_path>[...]</log_path>
  <log_only_on_failure>[...]</log_only_on_failure>
  <success_path>[...]</success_path>
  <failure_path>[...]</failure_path>
  <failure_notification_email>[...]</failure_notification_email>
</scheduling>
```

Configuration

- `log_path` *(optional)*: A log-file that carries the name of the workflow's filename appended with the execution date and time will be written to this folder.
- `log_only_on_failure` *(optional boolean; use true or false)*: Whether to write log files only when the workflow fails (true), or also when it succeeds (false, this is the default setting).
- `success_path` *(optional)*: When used in conjunction with a file-type connector, moves successfully processed files to this location
- `failure_path` *(optional)*: When used in conjunction with a file-type connector, moves files that resulted in a workflow failure to this location.
- `failure_notification_email` *(optional)*: When the workflow fails, a detailed error log will be sent to this e-mail address

7.3 Creating a Scheduled Task

When you load a workflow that has a scheduling configuration in the M-Port graphical user interface (GUI), the Scheduling menu will be shown. This menu displays the scheduling information for the workflow, allows you to open the Scheduled Tasks control panel applet, and allows you to create a new scheduled task for the workflow.

Creating a scheduled task from the M-Port GUI ensures the path settings of the command line executable to be properly set.

Workflows should be scheduled for the Administrator user of the domain or computer, and the checkbox for 'Run with highest priviliges' should be turned on.

7.4 M-Port Updates and Scheduled Tasks

Because M-Port uses a ClickOnce deployment method, each installation and update of the program will be

Because M-Port uses a ClickOnce deployment method, each installation and update of the program will be placed in a separate folder on your computer. These versions and updates are specific to the current user.

Because each update changes the installation folder, the location of the M-Port CLI changes with every update, too. That means that all workflows that have been scheduled need to be updated accordingly.

To counteract this behavior, M-Port creates a batch file in the default M-Port datastore location (see above); this batch file always points to the latest version of the M-Port CLI. Every first run of M-Port after an update takes care of updating the batch file's path.

8 Troubleshooting

More information on a particular error might be available in two files:

- `GBAe2apiErrors.txt`, in the *My Documents* folder. This file contains most Mamut API errors and might be a source for more information on the exact cause of a particular import failure.
- `_DbgMsgs.log`, in the Mamut data area, in the System0001 path (i.e., `\\SERVER\Mamut\Data\System0001\_DbgMsgs.log`). This file contains database log messages from Mamut.

8.1 No access to a particular company database

The API logs in using the first super user in the Mamut system database. If this user has no access to the company database you're trying to connect to, this will give an error message in `_DbgMsgs.log`, containing the username of the user that is used to log in. Using this username, give the user access to the database and connecting should work.

9 Changes

v3.3.14 - 2012-04-25

- Added: Field order.LineWarehouseID.

v3.3.13 - 2012-04-18

- Fixed: Multiple bugfixes related to parsing XML files.
- Fixed: AddLine processor to discard duplicate lines where appropriate.
- Added: Field person.Groups.

v3.3.12 - 2012-02-03

- Changed: Processor AddLines 'force_double' parameter.
- Added: Fields project.Budget*.

v3.3.11 - 2012-01-13

- Added: Processor AddLines.

v3.3.10 - 2012-01-06

v3.3.10 - 2012-01-06

- Added: Field timesheet.LineDepartmentID.
- Added: Field contact.OrderDiscountPercentage.

v3.3.9 - 2011-12-30

- Added: Processor KeyValue.
- Changed: Crystal Reports is no longer a prerequisite for installing M-Port. It can be downloaded separately from http://software.redant.net/m-port/CRRedist2008_x86.zip.

v3.3.8 - 2011-12-12

- Added: Fields order.Picked, order.Status.

v3.3.7 - 2011-11-18

- Added: Field order.DirectDebit.

v3.3.6 - 2011-10-21

- Added: Processor UserSelectDateRange.

v3.3.5 - 2011-09-30

- Added: Workflow descriptions.

v3.3.4 - 2011-07-05

- Added: Fields journal.DueDate, journal.Department, journal.DepartmentID, journal.Currency, journal.CurrencyID, journal.CurrencyRate.
- Added: Fields order.ServiceOrderText, order.ServiceOrderText2.

v3.3.3 - 2011-04-29

- Added: ExcelFormat for reading of XLSX files (Excel 2007 and later).
- Added: While writing to destination, M-Port now reports estimated time left until completion.
- Added: User startable Debug-mode (press F6 after startup).
- Added: Debug query executer in Debug menu.

v3.3.2 - 2011-04-08

- Added: Processor MultiLookUp for finding multiple database values in one query.

v3.3.1 - 2011-04-01

- Added: You can now insert products in structures using StructureProductID and QuantityInStructure.

- Added: Field product.StructureProductID, product.QuantityInStructure.

- Added: Field order.PickListText, order.DeliveryNoteText.
- Changed: Field product.ParentProductID is defined for variants, not structures.

v3.3 - 2011-03-25

- Added: Field contact.FactoringNumber.
- Added: Field order.DueDate.
- Added: Fields product.NextDeliveryDate, product.AvailableStock.
- Added: Field timesheet.LineToPayroll.
- Added: Some error messages from Mamut API are now rephrased to be more understandable for end users.
- Added: Windows-1252 encoding to File, Folder and FTP connectors.
- Added: A nice About screen :)
- Changed: GUI rearranged updating to Help menu in order to make room for additional functionality.
- Changed: Timestamp now contains a space instead of a - for the T marker.
- Fixed: XmlFormat handling of documents with non-standard root nodes.
- Fixed: XmlFormat handling of files starting with UTF8 byte order marker.
- Fixed: Handling of double conversion to string now inserts a starting 0 before the decimal point.
- Fixed: FixedFormat now properly normalizes string fields starting with a 0.
- Fixed: Joining of data tables.

v3.2.2 - 2010-12-17

- Added: Fields order.LineDeliveryDate, order.LineProject, order.LineProjectID.
- Added: Fields order.TotalWeight, order.TotalVolume, order.ItemAmount, order.PalletAmount, order.PickListText, order.DeliveryNoteText.
- Added: StringExtensions.InRange for use in Math processor.
- Added: QueryFormat now can execute a start_statement or end_statement before or after the line queries, respectively.

v3.2.1 - 2010-11-19

- Changed: MatchProcessor can now be terminated by using the X button.
- Added: Fields order.AmountDue and order.CreditnoteGenerated.

v3.2 - 2010-10-15

- Changed: Removed unnecessary converting to and from internal M-Port model.
- Changed: Optimized memory usage.
- Fixed: Field product.Costs (export).
- Fixed: Match processor now properly auto-selects, even if sorted.
- Added: MySQL connector now times out after 20 seconds.

v3.1.12 - 2010-10-08

- Added: Fields order.ResponseType and .ResponseTypeID can now be imported, too.

v3.1.11 - 2010-10-01

- Changed: Match processor now alphabetically sorts displayed data.
- Changed: Queue outputs now automatically proceed when used in a scheduled workflow.

v3.1.10 - 2010-09-17

- Added: Field order.AssociatedInvoiceID for credit notes.
- Changed: Normalize processor only outputs characters from a-z, A-Z, 0-9, space and hyphen (-).
- Changed: Reformatted changelog.
- Fixed: When creating an order, fixed Mamut API behaviour that puts in today's date when no delivery date is set.

v3.1.9 - 2010-08-27

- Updated: MatchProcessor can now ignore punctuation marks.
- Fixed: LookupProcessor now correctly displays lookup value when looking up a single field.
- Fixed: Workaround for Mamut API random OK errors when applying processors before Mamut destination.
- Fixed: Format FixedWidth with multiple input files.

v3.1.8 - 2010-07-15

- Changed: MatchProcessor can now check for matches for shortest string length only.
- Changed: MathProcessor now replaces " with ' for field replacements in expression and where.

v3.1.7 - 2010-07-09

- Added: Data type PurchaseOrder (export only).
- Added: Field order.UseLineGrossPrices for importing orders with prices that include VAT.
- Added: FieldPad for template editing.
- Fixed: Behaviour of field product.IsStockItem and .IsServiceItem (mutual exclusion).

v3.1.6 - 2010-07-02

- Fixed: Fields for person addresses.
- Added: Support for replacement of field codes used inside {temp.Subject} and {temp.Body} when used in Mail connector.

v3.1.5 - 2010-06-04

- Added: MatchProcessor can now be configured to show perfect matches.
- Added: MamutFormat can now create order lines without products (text only).

v3.1.4 - 2010-05-28

- Added: Query modificators for other datatypes than order.
- Added: Field person.IdentificationNumber.
- Fixed: Exporting of timesheet lines with empty intervals

- Fixed: Exporting of timesheet lines with empty intervals.
- Added: WebConnector now supports splitting of output.

v3.1.3 - 2010-05-21

- Added: FTPConnector encoding.
- Added: Field contact.IsMainOffice, .MainOffice, .MainOfficeCustomerID.
- Fixed: MatchProcessor now starts counting at -1 when matching empty databases.

v3.1.2 - 2010-05-14

- Added: Field order.LineQuantityDelivered.
- Added: Field product.TotalStock, fields for product accounts.
- Added: Percentage counter for most processors, and timer for long operations when in debug mode.
- Added: Log path and setting for writing logs on success, for scheduled workflows.
- Changed: MatchProcessor now has a setting to limit applicable match data, match_data_for.
- Changed: Creation of persons now check/creates the associated contact, allowing importing of contacts and persons in one workflow.
- Fixed: Splitting inside MailConnector.

v3.1.1 - 2010-05-06

- Added: Field product.VatRate and .PurchaseVatRate.
- Added: Field product.IndividualDiscount, .IndividualPriceWithoutDiscounts and .IndividualPrice.
- Fixed: Creation of variable path names in FileConnector.

v3.1.0 - 2010-05-03

This version is the first release that includes the M-Port CLI executable, for running workflows as scheduled tasks.

- Added: On first run after an update, CLI batch contents are updated.
- Added: Workflow scheduling menu for creating and viewing scheduled tasks.
- Added: GUI complety localized in Dutch, except for errors.
- Added: Processor Sum, for summing lines grouped on a specific field.
- Added: Processor Sort, for sorting columns.
- Added: Field order.NumberOfLines.
- Fixed: Joining of temp table in workflows that utilize multiple joined tables.
- Removed: Languages menu due to incompatibility with referenced libraries (M-Port Core, etc.); M-Port now always assumes the OS's language.

v3.0.3 - 2010-04-29

- Added: HKEY_LOCAL_MACHINE key to override current user settings.
- Changed: if both origin and destination connectors are database connectors, the preference for the processor connector is now the destination instead of the origin.

v3.0.2 - 2010-04-28

- Added: Datastore location can now be set on local machine key too.

- Changed: Support mailer not longer dependent on mailto-client.
- Changed: Implemented data table viewer as a modal dialog (finally!).
- Fixed: Improved MamutConnector error messages when no instance, system or company database has been selected.

v3.0.1 - 2010-04-27

- Changed: Added explanatory exceptions for common lookup functions (project, supplier id, employee id, contact id).
- Added: Field contact.OurReferenceID, AccountOnHold.
- Added: Field product.ContactKey and .CustomerID. There are not linked with contact table yet.
- Fixed: Query format now only transforms field syntax when retrieving data ({.} -> _).

v3.0.0 - 2010-04-23

This version is a huge rewrite of M-Port internals, in order to prepare for workflow scheduling.

- Changed: Moved datastore from isolated storage to a common application data folder.
- Changed: Exceptions thrown now show late bound class name instead of LazyType.
- Changed: Support mailer now includes log and Mamut API error log.
- Changed: Moved split directive from formats to connectors.
- Changed: All User* processors now cancel workflow execution when dismissed.
- Changed: MergeLinesProcessor can now merge lines of any datatype, not just orderlines.
- Changed: MySqlConnector now uses late binding to MySQL Connector/NET.
- Changed: QueryFormat now uses M-Port field notation instead of underscored fields.
- Changed: Data table viewer now displays nice column names.
- Changed: LookupProcessor now supports fields as default value.
- Added: FolderConnector for reading file batches.
- Added: Action Open in Notepad for File connector.
- Added: Datatype Journal can now be read as well as written.
- Added: Everywhere fields are used, {timestamp} is now replaced with the current sortable date and time.
- Added: Field order.ProductionDate.
- Added: Field journal.LineDescription, LineCostLayout, LineCostCenterGroup, LineCostCenter.
- Added: Fields project.Responsible and project.ResponsibleID.
- Added: Fields contact.ResponseType, ResponseTypeID, Currency, CurrencyID.
- Added: Fields person.Project, person.ProjectID, person.Memo.
- Fixed: Tab order for UserSelectRange processor.
- Fixed: MamutConnector max string length now > 256 characters.
- Fixed: M-Port now runs Mamut workflows on 64-bit systems (Mamut API requires target platform to be x86).

2010-03-30

- Fixed: Fields product.IsStockItem and product.IsServiceItem are now mutually exclusive.

2010-03-29

- Added: Fields for product settings.

2010-03-24

- Added: Field employee.Name.

2010-03-22

- Added: Datatype activity.

2010-03-17

- Added: Datatype currency. Fixed Mamut API bug regarding setting of exchange rates.

2010-03-16

- Changed: XmlFormat now allows for parsing of files with inline schemas.
- Added: WebConnector for communicating with SOAP compliant webservices.
- Changed: Fields contact.SyncID and person.SyncID are now strings.

2010-03-11

- Changed: Field contact.Project and contact.ProjectID now create a project link.

2010-03-04

- Added: now displays percentage complete for workflows that have Mamut as destination.

2010-03-03

- Improved: Parsing of 'first level has multiple nodes' XML template files.

2010-02-02

- Added: Fields product.Department, product.DepartmentID, product.Project, product.ProjectID.

2010-02-23

- Changed: Field order.FreightBillText can now be written to.
- Added: Field order.ShipmentNumber.

2010-02-22

- Improved: Parsing of complex XML template files.

2010-02-18

- Changed: Mamut sets Project to orderlines when Project has been assigned to order. Mamut API

- Changed: Mamut sets Project to deadline when Project has been assigned to order, Mamut API doesn't. M-Port now behaves like Mamut.

2010-02-17

- Fixed: Now properly shows error when access rights on isolated storage do not allow opening of workflow.

2010-02-16

- Added: Format Mamut now supports updating of orders, either by adding new lines or updating existing lines.
- Fixed: Format Field now correctly distinguishes between one and multiple regex line identifiers.

2010-02-08

- Changed: Processor Select no longer requires 'columns' node.

2010-02-04

- Changed: Removed workaround in AddressProcessor for country codes in the g_country table that were used twice. This is fixed in MBS 12.5.
- Changed: Workaround still applies for versions < 12.5.7897.

2010-02-01

- Added: Now returns latest Mamut API error from API error log file, when available.

2010-01-29

- Added: Query modificators (limit, order_by, where) for order export from Mamut.

2010-01-22

- Added: Orders now support person data on export.
- Changed: Empty person.* fields now no longer create empty company contacts on import.

2009-12-21

- Added: Fields product.CommodityCode, IndustryProductID, and Www.
- Changed: Fields product.WarehouseLocation and WarehouseLocationID now check for existing locations.
- Fixed: Fields product.Unit, UnitID, and Subgroup3.

2009-12-18

- Changed: Now compatible with MBS 12.5.
- Added: Fields project.Completion and CompletionID, and product.SupplierDeliveryTime

- Added: Fields project.CompletionID and CompletionID, and product.SupplierDeliveryTime.
- Changed: product.WarehouseLocation is now automatically created if new.

2009-12-15

- Added: Fields product.WarehouseLocation and product.WarehouseLocationID.
- Changed: Most standard register fields now are check created instead of looked up.

2009-12-08

- Changed: Processor MergeLine now can merge product bundles automatically.

2009-12-07

- Added: Field order.LineType.
- Added: Fields product.SurchargeID, product.SurchargeGroup, product.SurchargeDescription.
- Added: Model table OrderlineMatrix (10x10 matrix with headings).
- Fixed: Now throws error when setting store link values when no warehouse ID is specified.
- Changed: When setting a new warehouse for a product, stock is assumed 0 when not supplied.

2009-12-03

- Added: FieldFormat.
- Added: Field order.LineTracingNumber.

2009-12-02

- Added: XSL transformations added to XML format.

2009-11-26

- Changed: Error dialog now hides details by default.
- Fixed: Order lines with missing information now behave as expected in Mamut.

2009-11-25

- Added: Field product.OverrideSalesCostPrice.

2009-11-24

- Added: MySqlConnector. Requires installation of MySQL Connector/Net (http://dev.mysql.com/downloads/connector/net/).
- Changed: SqlConnector is now compatible with QueryFormat's PutContents.

2009-11-20

- Fixed: FileConnector when both regex replace and multiline regex replace were used.

2009-11-19

- Added: Field contact.CreditLimit.

2009-11-12

- Added: Processor MergeLinesProcessor. Can merge multiple lines and sum fields therein.
- Changed: Processor SelectProcessor now accepts {} instead of backticks for fields.

2009-11-11

- Added: Processor MathProcessor. Can perform calculations on input data.

2009-11-09

- Changed: QueryFormat can now put contents (e.g., write custom update queries). Queue is enabled for this format.
- Fixed: Workflow/customer names with ampersands are now properly shown in workflows menu.

2009-11-05

- Added: Fields contact.DeliveryMethod, contact.DeliveryMethodID, contact.DeliveryTerms, contact.DeliveryTermsID.

2009-10-30

- Changed: MailConnector now detects empty e-mail addresses.

2009-10-29

- Added: FileConnector and FTPConnector now support multiline regex replace.

2009-10-28

- Added: Queue window for processing multiple (split) exports.
- Added: MailConnector now accepts queue adding and processing.
- Changed: OneTimeProcessor does not store values that generated errors in queue.

2009-10-21

- Added: Field product.RecommendedSalesPrice (excl. VAT!)
- Added: Field product.Inactive -- with new behaviour (exports only active products, unless this field is included).
- Changed: Processor UserVerify now shows total of displayed rows instead of total data rows.
- Changed: Field product.SupplierPrice now correctly calculates and sets price in local currency.
- Removed: DummyFormat, DummyConnector.

2009-10-20

- Added: Field product.PictureLink2.
- Added: Registry keys to determine customer and debug mode.
