# RTF Templates Guide - revision V7R2

Table of Contents

# 1 Introduction

### Templates in Translation Office 3000, Version 9.0

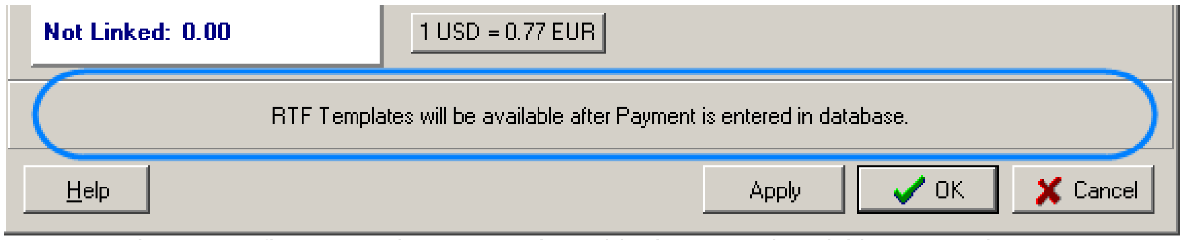
*Template* is an *RTF* ("Rich Text Format") file stored in your Translation Office 3000 setup folder and used as a 'template' when *saving work flow documents in RTF or PDF files*. RTF files can be opened with most text editors, including MS Word.

### Saving a document using a Templates

The following documents can be saved as printable RTF, PDF and DOC files using their own templates:

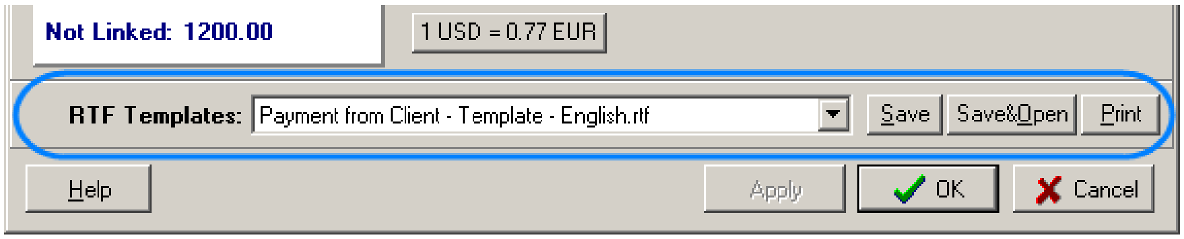
- Invoices to clients
- Payments from clients summaries
- Project and client job summaries
- Quotes to clients
- Prices

Documents can be saved using windows **Edit Invoice**, **Edit Project** and so on, as well as **Prices** tabs of **Client** window, and **General Prices for Clients** and window.
Any window which has RTF Templates section can be used to save its data in a printable document. Until the data which you are editing is saved in your database, RTF Templates section will look like this:



RTF Templates will be available after Payment is entered in database.

To save a document, first apply changes made to this document by clicking **Apply** button. Once the document has been saved in database, RTF template controls for it will become active:

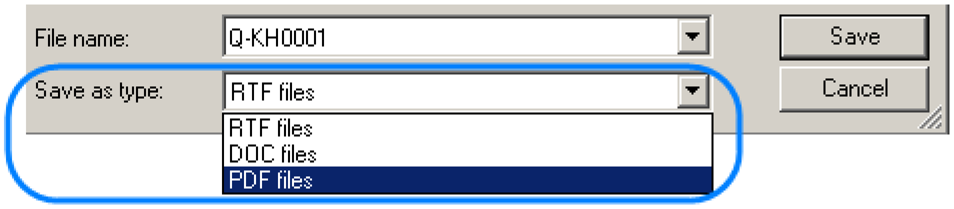


RTF Templates: Payment from Client - Template - English.rtf

- Use **RTF Templates** drop-down list to select the template which must be used for saving this document in an RTF file.

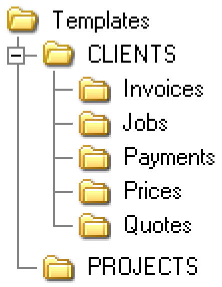
- Click **Save** button to save template in the respective payment folder.
- Click **Save&Open** button to open this payment in your default text editor immediately after saving it.
- Click **Print** button to quickly send this payment to printer.

**Note:** You can choose to save the output file in *RTF*, *PDF* or *DOC* format by selecting this format in **Save As** window with the help of **Save as type** drop-down list.



## Template files

Each document type (like *invoices*, *purchase orders*, *quotes*, and so on) uses templates stored in specific folder, named by the document type, which templates it stores:



By editing template overlay and format, you edit overlay and format of all the documents which will be saved with it. There can be a number of templates to choose from for one document type.

## Editing templates

Templates can be opened for editing using the **Templates** tab of Translation Office 3000, Version 9.0 **Personal Settings.** You can access and manage Translation Office 3000, Version 9.0 RTF templates used for issuing *invoice*, *quotes* and saving other kinds of information from Translation Office 3000, Version 9.0 database.

**How to:**

1. To open **Templates** tab click **Personal** command from **Settings** menu of Translation Office 3000, Version 9.0 and select **Templates** tab.
2. Navigate to the folder of the document, which templates you wish to edit.
3. Open required template file with text editor (MS Word or any other editor supporting RTF format).
4. After making all the necessary changes, save **Template** in the same or new RTF file of the same

folder.

**Note:** Templates folders can also be opened with regular file browser, like Windows Explorer.

## Template Variables

Variable is a certain symbol combination which is entered in template files. When a document is saved using template, Translation Office 3000, Version 9.0 recognizes the variable and inserts required data in the resulting document.
Variables are indicated with "\" (backslash) symbols in the beginning and in the end,
➡ **Example:** \CLIENT_NAME\, \CLIENT_STREET1\, \CLIENT_PHONE1\ etc.
Each variable represents a portion of data which will be inserted in its place when the document is saved.

**Note:** Template can be edited without changing the variables wording and outlay (fonts and colors can still be changed).
Advanced users can learn about templates variables; add/delete them as necessary, customizing templates to the maximum.

See also:
Logic of Templates

# 2 Logic and syntax of templates

## Variables

All variables and commands can be identified by \ (backslash) symbol in the beginning and in the end.

**Example:**

Below you can see \CLIENT_NAME\, \CLIENT_STREET1\ and \CLIENT_PHONE1\ variables:

**To:** \CLIENT_NAME\

**Address:** \CLIENT_STREET1\

**Phone:** \CLIENT_PHONE1\

When you use this template by clicking on **Save** or **Print** in some dialog box, \CLIENT_NAME\ variable will be replaced by name of currently selected *client*, \CLIENT_STREET1\ — by *client's street address* and \CLIENT_PHONE1\ — by *client's phone number*. The final output will look like this:

**To:** XYZ Company

**Address:** Elm Street, 1

**Phone:** +1 212 898 11 31

**Note:** All template variables can be sorted into 7 groups: Common template variables, Project template variables, Contacts template variables, Quotes template variables, Client jobs template variables, Invoice template variables and Payments template variables. Common template variables can be used in any type of template. All other template variables can be used only in certain types of templates. More about the use of certain variables in certain templates can be read in the specific topics about groups of variables.

## Data Scan commands

These commands are used to create tables in which number of rows is determined by the number of records in database.

**Example:**

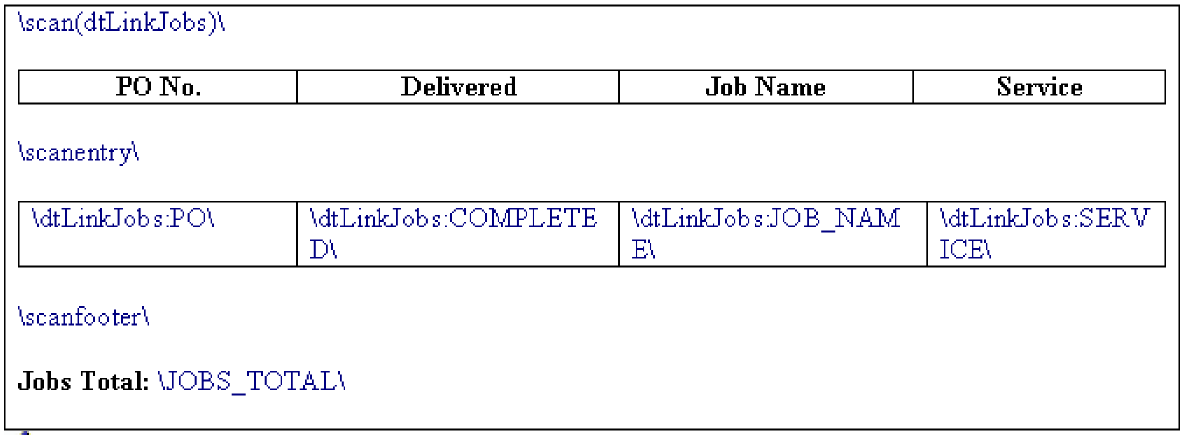
1. Begin by entering \scan(dtLinkJobs)\ command. This must not necessarily be dtLinkJobs, depending on the template, this can be:

- \scan(dtLinkJobs)\ — in *invoices* templates, to display jobs included in invoice.
- \scan(dtLinkInvoices)\ — in *payment* templates to display invoices linked to the payment.
- \scan(dtLinkPayments)\ — in *invoice* templates to indicate linked sums.

2. Insert table header between \scan(dtLinkJobs)\ and \scanentry\.

3. Insert one row of data-columns after \scanentry\. Every variable in table must be preceded with dtlink command. For instance if the table begins with \scan(**dtLinkJobs**)\, each variable must have this link included: **\dtLinkJobs**:CJOB_NAME\, **\dtLinkJobs**:CJOB_COMPLETED\, and so on.
4. Insert \endscan\ after data-columns.

**Example:**



\scan(dtLinkJobs)\

| PO No. | Delivered | Job Name | Service |
|---|---|---|---|

\scanentry\

| | | | |
|---|---|---|---|
| \dtLinkJobs:PO\ | \dtLinkJobs:COMPLETED\ | \dtLinkJobs:JOB_NAME\ | \dtLinkJobs:SERVICE\ |

\scanfooter\

**Jobs Total:** \JOBS_TOTAL\

**Note:** Use noeof to hide table's header and footer if the table body appears to be empty. For instance, if your *invoice* includes only *jobs* and no *payments*, the header and footer for *payments* will not be saved/printed.

**Example:**

\scan(DTLINKPAYMENTS),noeof\

Command \scan(dtLinkJobs)\ will make the parser scan (go from first record to the last one) all selected data in **Jobs** table (in this particular case all *jobs* included in the *invoice*) and list them in the resulting file.
Text between \scan(dtLinkJobs)\ command and \scanentry\ command is table header.
Text between \scanfooter\ and \endscan\ commands will be considered table footer and will only be displayed once at the end of this table.
Text between \scanentry\ and \scanfooter\ commands is a draft determining the format of each line in this table between footer and header. It includes variables from columns which must be listed in the table. In example above these are:

- \DTLINKJOBS:CJOB_PONUMB\ – PO Number.
- \DTLINKJOBS:CJOB_COMPLETED\ – Date of job delivery.
- \DTLINKJOBS:CJOB_NAME\ – Name of the job.
- \DTLINKJOBS:SERV_NAME\ – Service provided.

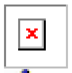
**Note:** Numerical values can be rounded up by fnum command. This command will round up the value to specified number of digits after decimal point (2 digits in the example below):

**Example:**

**Job Total:** \fnum(dtLinkJobs:TOTAL, 2)\

**Note:** If the command \scanentry\ is placed before the header (see the example below), then the header will be displayed over every row in the table.

**Example:**



**Note:** The full list of dtLinkJobs, dtLinkInvoices and dtLinkPayments variables can be found in corresponding topics of this chapter.

## Condition checking

Templates can react to certain varying conditions and produce separate output for each of the conditions. The syntax of if command is the following:

\IF(**condition**)\ **Reaction** \ENDIF\.

**Example:**

If *Discount 1* is applied, display *discount name*, *discount value* and *subtotal*. The code is as follows:

```
\IF(DISCOUNT1)\
\DISCOUNT1NAME\: \DISCOUNT1VALUE\
Subtotal: \AFTERDISCOUNT1\\endif\
```

This will make the template check if discount 1 is applied, and if so — display data between \IF(...)\ and \ENDIF\ commands.

- DISCOUNT1 is a logical variable, i.e. it can only have one of two values: either *true* or *false*. In this particular case, Translation Office 3000, Version 9.0 assumes that DISCOUNT1 is *true* if first discount is present. If no first discount has been applied, then this value is assumed as *false*.
- When \IF(DISCOUNT1)\ is encountered in template, parser checks DISCOUNT1 logical value, and if it is true, runs the code below this command, until \endif\ is encountered, which instructs parser to stop. If DISCOUNT1 is false, everything until \endif\ command is skipped.
- In this particular case, without \IF(DISCOUNT1)\ command, the parser would output empty string instead of \DISCOUNT1NAME\: \DISCOUNT1VALUE\ and **Subtotal:** text in case there is no discount. In case \IF(DISCOUNT1)\ command is used, Translation Office 3000, Version 9.0 will

adhere to *IF* command and skip unnecessary text entirely in case no discounts are applied.

## Arithmetic calculations syntax

Arithmetic can be performed in templates with the following syntax:

```
 \((JOBS_TOTAL+100-10)*10)/2\
```

- The whole expression must be included in backslashes;
- All arithmetic operations must be defined by the following symbols + - / *;
- The brackets inside backlashes must be positioned by the rules of arithmetic.

## Logical operations syntax

Logical operations: && (and), || (or), ! (not),  can be performed in templates with the following syntax:

```
\IF( (table1:field1>b+1) || (table1:field1=0) )\

............

\ENDIF\
```

# 3 Common template variables

There are four categories of common variables which can be used in all templates:

- User information variables;
- Current date variables;
- Client information variables;
- Client currency variable.

## User information variables

| VARIABLE: | DESCRIPTION: |
|---|---|
| \COMPANY_NAME\ or \USER_NAME\ | Name of **registered company** (i.e. your company name) taken from Translation Office 3000, Version 9.0 license key, cannot be modified within program. |
| \COMPANY_PAYMENT_TERMS\ | Company’s payment terms (your company’s payment terms of freelance experts). |
| \COMPANY_PAYMENT_TERMS_NOTES | Additional notes you have entered for payment terms for freelancers. |
| \COMPANY_CURRENCY\ | Company's base currency |
| \CURRENT_USER_NAME\ | **Name** of the logged-in user (as specified in the Users and Access). |
| \CURRENT_USER_POSITION\ | **Position** of the logged-in user (as specified in the Users and Access**)**. |

## Date variables

| VARIABLE: | DESCRIPTION: |
|---|---|
| \DATE\ | Date in short format (for example, 10/4/2006) |
| \DATE_LONG\ or \LONGDATE\ | Date in long format (for example, Monday, October 04, 2006) |

## Client information variables

This set is available in all templates except Profile templates when used in **Master Profile**:

| VARIABLE: | DESCRIPTION: |
|---|---|
| \CLIENT_NAME\ | Name of client, related to the document. |
| \CLIENT_CODE\ | Code of client, related to the document. |
| \CLIENT_CURRENCY\ | Currency of client, related to the document. |
| \CLIENT_MINFEE\ | Minimum fee of client, related to the document. |
| \CLIENT_PAYMENT_TERMS\ | Payment terms of client, related to the document. |
| \CLIENT_PAYMENT_TERMS_NOTES\ | Additional notes on payment terms of client, related to the document. |

| VARIABLE: | DESCRIPTION: |
|---|---|
| \CLIENT_ADDRESS\ | Whole address of the client. This variable has pre-defined order. If address format is different in your country, you can use separate address items (variables below) to include address into your customized template. |
| \CLIENT_STREET1\ | Street address of client document was produced for. |
| \CLIENT_STREET2\ | Street address 2 (if available) of client document was produced for. |
| \CLIENT_CITY\ | City of client, related to the document. |
| \CLIENT_STATE\ | State of client, related to the document. |
| \CLIENT_COUNTRY\ | Country of client, related to the document. |
| \CLIENT_ZIP\ | ZIP of client, related to the document. |
| \CLIENT_EMAIL1\ | Email of client, related to the document. |
| \CLIENT_EMAIL2\ | Email 2 (if available) of client, related to the document. |
| \CLIENT_PHONE1\ | Phone number of client, related to the document. |
| \CLIENT_PHONE2\ | Phone number 2 (if available) of client, related to the document. |
| \CLIENT_PHONE3\ | Phone number 3 (if available) of client, related to the document. |
| \CLIENT_PHONE4\ | Phone number 4 (if available) of client, related to the document. |
| \CLIENT_FAX\ | Fax number of client, related to the document. |
| \CLIENT_WEB\ or \CLIENT_WWW\ | Web-site address of client, related to the document. |

| VARIABLE: | DESCRIPTION: |
|---|---|
| \CLIENT_MINFO\ | Application information of client, related to the document. |
| \CLIENT_MWEB\ or \CLIENT_MURL\ | URL (Web tab address) for application submission / information. |
| \CLIENT_INFO\ | General Information about the client, related to the document. |
| \CLIENT_VATNUM\ | VAT Number of client, related to the document. |

**Note:** \CLIENT_ADDRESS\ variable has pre-defined order. If address format is different in your country, you can use separate address items to include address into your customized template.

Use comma-separated variables \CLIENT_STREET1_C\, \CLIENT_STREET2_C\, \CLIENT_CITY_C\, \CLIENT_STATE_C\, \CLIENT_COUNTRY_C\, \CLIENT_ZIP_C\ is necessary if you would like parts of address to be separated by commas. You can as well insert commas directly into template but in this case unnecessary commas may appear even if information in some variables (like second line of street address) is empty.

## Client currency variable

\CLIENT_CURRENCY\ variable always contains *client currency* of currently selected *client*.

See also:
Logic of Templates

# 4 Date and time functions

In most cases database stores complete date and time. Certain commands can be used to customize the format of output date and time data (you may want to output only the *day* of the *week* or only the time etc.).

The following date and time functions can be used in all templates:

- fmdt
- wd
- date
- time

**Note:** These functions return value according to Regional and Language Options settings in your system. These options can be changed wit the help of your Windows Control Panel.

**Example:**

The same variable will be displayed in different way depending on the function used:

| FUNCTION: | VARIABLE VALUE: | FUNCTION APPLIED: |
|---|---|---|
| fmdt | 9/20/06 6:00 PM | Wednesday, September 20, 2006 6:00 PM |
| | 9/20/06 | Wednesday, September 20, 2006 |
| wd | 9/20/06 6:00 PM | Wednesday |
| | 9/20/06 | Wednesday |
| date | 9/20/06 6:00 PM | 09/20/06 |
| | 9/20/06 | 09/20/06 |
| time | 9/20/06 6:00 PM | 6:00 PM |
| | 9/20/06 | (empty row) |

Date and time functions can be used in all templates and can be applied to:

- All variables from datasets which return date and time.
- All variables from the following table:

| VARIABLE: | TYPE: | DESCRIPTION: |
|---|---|---|
| \PROJECT_DATE_STARTED\ | Project template variable | Date when the *project* had been started. Format: 9/20/2006. |
| \PROJECT_DATE_DEADLINE\ | Project template variable | *Project* deadline. Format: 9/20/2006 . |
| \PROJECT_DATE_COMPLETED\ | Project template variable | Date of the *project* completion. Format: 9/20/2006. |
| \ASSIGNED\ | Client, Corporate Experts, Freelancer Jobs template variable | Date when the *job* was assigned. Format: 9/20/2006. |
| \DEADLINE\ | Client, Corporate Experts, Freelancer Jobs template variable | *Job* deadline. Format: 9/20/2006 |
| \COMPLETED\ | Client, Corporate Experts, Freelancer Jobs template variable | Completion date. Format: 9/20/2006 . |
| \DONE\ | Client Jobs template variable | Completion date. Format: 9/20/2006 . |
| \START\ \ESTSTART\ | Quotes template variable | Date assigned. Format: 9/20/2006. |
| \COMPLETION\ \ESTCOMPLETION\ | Quotes template variable | Deadline date. Format: 9/20/2006. |

## Syntax

Date and time functions must be added to the variable in the following way:

\function(VARIABLE)\

**Example:**

To add wd function to \ASSIGNED\ variable from the *client jobs* template, change the variable syntax in the following way:

\wd(ASSIGNED)\

The result will be the day of the week, when the *job* was assigned (e.g. Wednesday).

See also:

Logic of templates

# 5 Project template variables

These variables can be used in project templates, as well as in any job or job assignment templates.

| VARIABLE: | DESCRIPTION: |
|---|---|
| \PROJECT_NAME\ | Name of *project*. |
| \PROJECT_CODE\ | *Project* code. |
| \PROJECT_CLIENT_NAME\ | *Client* of this *project*. |
| \PROJECT_CLIENT_CODE\ | *Client* reference number. |
| \PROJECT_INFO\ | Information about the *project*. |
| \CLIENT_PM_NAME\ | Client *project manager* |
| \PROJECT_DATE_STARTED\ | Date when *project* was started. Format: 10/4/2006. |
| \PROJECT_DATE_DEADLINE\ | *Project* deadline. Format: 10/4/2006. |
| \PROJECT_DATE_COMPLETED\ | Date of *project* completion. Format: 10/4/2006. |
| \PROJECT_DATE_STARTED_LONG\ | Date when *project* was started. Format: Monday, October 04, 2006. |
| \PROJECT_DATE_DEADLINE_LONG\ | *Project* deadline. Format: Monday, October 04, 2006. |
| \PROJECT_DATE_COMPLETED_LONG\ | Date of *project* completion. Format: Monday, October 04, 2006. |

See also:

Logic of templates

# 6 Contacts template variables

These variables are used in any template mentioning client's details.

| VARIABLE | DESCRIPTION |
|---|---|
| \SALUTATION\<br>\SAL\<br>\CONTACT_SALUTATION\ | For example: “Mr.”, “Ms.”, “Mrs.” etc. |
| \CONTACT_TITLE\<br>\TITLE\ | *Contact* title. |
| \CONTACT_NAME\<br>\PM_NAME\<br>\ATTENTION\ | *Contact* name. |
| \CONTACT_EMAIL1\ | *Contact* email address. |
| \CONTACT_EMAIL2\ | *Contact* second email address. |
| \CONTACT_PHONE1\ | *Contact* phone number. |
| \CONTACT_PHONE2\ | *Contact* second phone number. |
| \CONTACT_FAX\ | *Contact* fax number. |
| \CONTACT_NOTES\ | *Contact* notes. |

See also:

Logic of Templates

# 7 Quotes template variables

Using variables of Quote templates you can construct templates either for your reference or for sending to *client* by email or fax.

| VARIABLE | DESCRIPTION |
|---|---|
| \QUOTE_NAME\ | Possible *quote* name. |
| \QUOTE_CODE\ \CODE\ | *Quote* code. |
| \SERVICE\ | *Service* name. |
| \REQUEST\ | Request for *quote*. |
| \ANSWER\ | Answer to request for *quote*. |
| \VOLUME\ | *Quote* volume. |
| \PRICING\ \TYPE\ | *Quote* type. |
| \PRICE\ | *Quote* price. |
| \UNITS\ | *Quote* units. |
| \TOTAL\ | *Quote* total. |
| \DATE_SENT\ \SENT\ | Date. Format: 10/4/2006. |
| \DATE_SENT_LONG\ \LONGSENT\ | Date sent. Format: Monday, October 04, 2006. |
| \START\ \ESTSTART\ | Date assigned. Format: 10/4/2006. |
| \START_LONG\ \LONGESTSTART\ | Date assigned. Format: Monday, October 04, 2006. |
| \COMPLETION\ \ESTCOMPLETION\ | Deadline date. Format: 10/4/2006. |
| \LONG_COMPLETION\ \LONGESTCOMPLETION\ | Deadline date. Format: Monday, October 04, 2006. |
| \COUNT_NOTES\ | CATCount notes. |

| | |
|---|---|
| \STATUS\ | Status of *quote* (*unknown, accepted, rejected*) |

See also:

Logic of Templates

# 8 Client jobs template variables

Variables for templates, used to save data in **New/Edit Job** dialog box can be used to construct document templates either for your reference during work process or for confirming job details to client.

| VARIABLE | DESCRIPTION |
|---|---|
| \JOB_NAME\ | *Job* name. |
| \JOB_CODE\<br>\CODE\ | *Job* code. |
| \PO_CODE\<br>\PO\ | Purchase order *client* issued for this *job*. |
| \CLIENT_REF\ | *Client* reference number in accounting system of *client*. |
| \SERVICE\ | *Service* name. |
| \INSTRUCTIONS\ | *Job* instructions. |
| \WORK_NOTES\ | Work notes. |
| \VOLUME\ | *Job* volume. |
| \TYPE\ | *Job* type (for example: per unit, flat fee, free) |
| \PRICE\ | *Job* price. |
| \UNITS\ | *Job* units. |
| \TOTAL\ | *Job* total. |

| VARIABLE | DESCRIPTION |
|---|---|
| \ASSIGNED\ | Date when *job* was assigned. Format: 10/4/2006. |
| \ASSIGNED_LONG\<br>\LONGASSIGNED\ | Date when *job* was assigned. Format: Monday, October 04, 2006. |
| \DEADLINE\ | *Job* deadline. Format: 10/4/2006. |
| \DEADLINE_LONG\<br>\LONGDEADLINE\ | *Job* deadline. Format: Monday, October 04, 2006. |
| \COMPLETED\<br>\DONE\ | Completion date. Format: 10/4/2006. |

| | |
|---|---|
| \COMPLETED_LONG\ <br> \LONGCOMPLETED\ | Completion date. Format: Monday, October 04, 2006. |
| \COUNT_NOTES\ | CATCount or AnyCount notes. |
| \INVOICE_CODE\ <br> \INVOICE\ | *Invoice* code. |
| \INVOICE_GLOBAL_CODE\ <br> \INV_GLOBAL\ | *Invoice* global code. |

See also:

Logic of Templates

# 9 Invoice template variables

## Basic invoice variables

The following variables can be entered anywhere in your invoice template.

| VARIABLE | DESCRIPTION |
|---|---|
| \STATUS\ | Invoice status (For example: “Expected within 30 days”, “Settled 5 days earlier” etc.) |
| \DATE_ DUE\<br>\SETTLEMENT_DATE\<br>\DUE_DATE\ | Date when invoice is due. Format: 10/4/2006. |
| \DATE_DUE_LONG\<br>\SETTLEMENT_LONGDATE\<br>\DUE_DATELONG\ | Date when invoice is due. Format: Monday, October 04, 2006. |
| \INVOICE_DATE\<br>\INV_DATE\ | Date invoice sent. Format: 10/4/2006. |
| \INVOICE_DATE_LONG\<br>\INV_LONGDATE\ | Date invoice sent. Format: Monday, October 04, 2006. |
| \INVOICE _CODE\<br>\INV_CODE\ | *Invoice* code |
| \INVOICE _GLOBAL_CODE\<br>\INV_GLOBAL\<br>\INV_GLOBALLONG\ | *Invoice* global code**.** |
| \INVOICE_TOTAL\<br>\INV_TOTAL\ | *Invoice* total. |
| \JOBS_TOTAL\ | *Jobs total*. |
| \TAX1\<br>\TAX\ | Used in algorithms (If *tax 1 exists*, then *True*, if the *tax 1 does not exist*, then *False*) |
| \TAX1_NAME\<br>\TAX1NAME\<br>\TAXNAME\ | *Tax 1* name. |
| \TAX1_VALUE\<br>\TAX1VALUE\<br>\TAXVALUE\ | *Tax 1* value. |
| \TAX2\ | Used in algorithms (If *tax 2 exists*, then *True*, if the *tax 2 does not exist*, then *False*) |

| | |
|---|---|
| \TAX2_NAME\<br>\TAX2NAME\ | *Tax 2* name. |
| \TAX2_VALUE\<br>\TAX2VALUE\ | *Tax 2* value. |
| \DISCOUNT1\<br>\DISCOUNT\ | Used in algorithms (If *discount 1 exists*, then *True*, if the *discount 1 does not exist*, then *False*) |
| \DISCOUNT1_NAME\<br>\DISCOUNTNAME\<br>\DISCOUNT1NAME\ | *Discount 1* name. |
| \DISCOUNT1_VALUE\<br>\DISCOUNT1VALUE\<br>\DISCOUNTVALUE\ | *Discount 1* value. |
| \DISCOUNT2\ | Used in algorithms (If *discount 2 exists*, then *True*, if the *discount 2 does not exist*, then *False*) |
| \DISCOUNT2_NAME\<br>\DISCOUNT2NAME\ | *Discount 2* name. |
| \DISCOUNT2_VALUE\<br>\DISCOUNTVALUE\ | *Discount 2* value. |
| \TAXES\ | Used in algorithms (If *any tax is set*, then *True* , if there are *no taxes set*, then *False*) |
| \DISCOUNTS\ | Used in algorithms (If *any discount is set*, then *True*, if there are *no discounts set*, then *False*) |
| \AFTER_DISCOUNT1\<br>\AFTERDISCOUNT1\<br>\AFTERDISCOUNT\ | Total after *Discount 1* has been applied. |
| \NET_JOBS_TOTAL\<br>\NETJOBSTOTAL\ | *Jobs* total with discounts. |
| \AFTER_TAX1\<br>\AFTERTAX1\ | Total after *Tax 1* has been applied. |
| \INV_DUE\<br>\INVOICE_DUE\ | Balance due. |
| \INV_PAID\<br>\INVOICE_PAID\ | Sum paid for this *invoice*. |

| | |
|---|---|
| \INV_IS_PAID\ | Used in algorithms (If the *invoice* is paid then *True*, if invoice is not paid, then *False*) |
| \BEFORE_ADJUSTMENTS\ | *Invoice* total, excluding *adjustments* |
| \ADJUSTMENTS_VALUE\ | Value of the *adjustments* |
| \ADJUSTMENTS_DESCR\ | Description of the *adjustments* |
| \INVOICE_PAYMETHOD\ | Invoice payment method. |
| \INVOICE_PAYMETHOD_DESCR\ | Payment method description. |

## Jobs variables in invoice

The following variables refer to jobs and added to invoice through DTLINKJOBS function.

| **DATASET WITH COLUMN NAME** | **DESCRIPTION** |
|---|---|
| \DTLINKJOBS:CJOB_PONUMB\ | *Client* PO of the *job* |
| \DTLINKJOBS:CJOB_NAME\ | *Client job name* |
| \DTLINKJOBS:CJOB_ASSIGNED\ | *Assigned* date of client job |
| \DTLINKJOBS:CJOB_DEADLINE\ | *Deadline* date of client job |
| \DTLINKJOBS:CJOB_ISCOMPLETED\ | *Completed* (Boolean: True/False) |
| \DTLINKJOBS:CJOB_COMPLETED\ | *Completed* date of client job |
| \DTLINKJOBS:CJOB_PRICE\ | *Unit price* of client job |
| \DTLINKJOBS:CJOB_VOLUME\ | *Volume* of client job |
| \DTLINKJOBS:CJOB_FEE_KIND\ | *Pricing* of client job (i.e. per unit, flat fee) |
| \DTLINKJOBS:CJOB_RATE\ | *Exchange rate* of client job |
| \DTLINKJOBS:CJOB_TOTAL\ | *Job total* of client job |
| \DTLINKJOBS:CJOB_INSTRUCTION\ | *Instructions* of client job |
| \DTLINKJOBS:CJOB_WORKNOTES\ | *Work notes* of client job |
| \DTLINKJOBS:CJOB_COUNTNOTES\ | *Count notes* CATCount notes of client job |
| \DTLINKJOBS:SERV_NAME\ | *Service* type of client job |
| \DTLINKJOBS:UNIT_NAME\ | *Units* of client job (i.e. words, lines, CAT text) |
| \DTLINKJOBS:PROJ_CODE\ | *Project code* of client job |
| \DTLINKJOBS:CJOB_CODE\ | *Job code* of client job |
| \DTLINKJOBS:CJOB_CLCODE\ | Client Ref. value of client job |
| \DTLINKJOBS:CCON_NAME\ | *Client PM* of client job |

## Payments variables in invoice

The following variables refer to payments and added to invoice through DTLINKPAYMENTS function.

| DATASET WITH COLUMN NAME | DESCRIPTION |
|---|---|
| \DTLINKPAYMENTS: CPAYM_CODE\ | *Payment* code. |
| \DTLINKPAYMENTS: CPAYM_DATE\ | Date of *payment*. |
| \DTLINKPAYMENTS: CPAYM_TOTAL\ | Total paid. |
| \DTLINKPAYMENTS: LINK_SUM\ | Amount linked with *invoice.* |

See also:

Logic of Templates

# 10 Payments template variables

## Basic payments template variables

The following variables can be used to create payment templates

| VARIABLE | DESCRIPTION |
|---|---|
| \PAYMENT_CODE\ <br> \CODE\ | *Payment* code. |
| \PAYMENT_DATE\ <br> \PDATE\ | Date of *payment*. Format: 10/4/2006. |
| \PAYMENT_DATE_LONG\ <br> \PLONGDATE\ | Date of *payment.* Format: Monday, October 04, 2006. |
| \PAYMENT_NOTES\ | Notes of *payment*. |
| \TOTAL_PAID\ <br> \TOTAL\ | Total paid. |
| \NOT_LINKED\ | Amount not linked with *invoices*. |
| \LINKED\ | Amount linked with *invoices*. |
| \IS_LINKED\ | Used in algorithms (If *payment* is linked with *invoice* = True, if it's not = False) |
| \PAYMENT_NOTES\ <br> \NOTES\ | Payment notes |

## Linked invoice variables

These variables can add information from linked invoices to payment template

| DATASET WITH COLUMN NAME | DESCRIPTION |
|---|---|
| \DTLINKINVOICES:IDATE\ | Linked invoice date. |
| \DTLINKINVOICES:ICODE\ | Linked invoice code. |
| \DTLINKINVOICES:GNUMB\ | Linked invoice global code. |
| \DTLINKINVOICES:TOTAL\ | Linked invoice total. |
| \DTLINKINVOICES:OTHER\ | The part of the linked invoice total covered by other payments. |
| \DTLINKINVOICES:ADJUST\ | The sum of the phantom payment of the linked invoice. |
| \DTLINKINVOICES:LINKED\ | The part of the current payment total linked to the invoice. |

| | |
|---|---|
| \DTLINKINVOICES:BALANCE\ | *Balance Due* of the linked invoice. |
| \DTLINKINVOICES:DATEDUE\ | Linked invoice due date. |

See also:

Logic of Templates

# 11 Advanced commands and functions

**Format of IF-ELSIF-ELSE-ENDIF construction is:**

\If**(Condition1)**\
**Reaction1**
\elsif**(Condition2)**\
**Reaction2**
\else\
**Reaction3**
\endif\

**Note:** \If\ and \endif\ are the mandatory commands in this construction, \elsif\ and \else\ are an optional commands. Condition must be a variable with boolean value. Such variables can have only two values: True or False.

**Example:**

| **Invoice template** | **Explanation** |
|---|---|
| \If(INV_IS_PAID)\<br>**Invoice is paid**<br>\else\<br>**Invoice is not paid**<br>\endif\ | If the invoice is paid(INV_IS_PAID=True),then<br>**Invoice is paid**<br>is displayed in produced invoice, if invoice is not paid( INV_IS_PAID=False), then<br>**Invoice is not paid**<br>is displayed in produced invoice. |

**or**

| **Invoice template** | **Explanation** |
|---|---|
| \scan(dtLinkJobs)\<br>……..<br>\scanentry\<br>……..<br>\If(INV_IS_PAID)\<br>**Invoice is paid**<br>\elsif(DTLINKJOBS:CJOB_ISCOMPLETED=true)\<br>**Invoice is not paid**<br>\else\<br>**Job is not completed**<br>\endif\<br><br>\scanfooter\<br>……..<br>\endscan\ | If the invoice is paid(INV_IS_PAID=True),then<br>**Invoice is paid** is displayed in produced invoice,<br>if invoice is not paid(INV_IS_PAID=False), then, if ( DTLINKJOBS:CJOB_ISCOMPLETED=true),<br>then **Invoice is not paid** is displayed in a produced invoice,<br>if invoice is not paid(INV_IS_PAID=False) and if ( DTLINKJOBS:CJOB_ISCOMPLETED=false),then<br>**Job is not completed** is displayed in a produced invoice. |

**IIF function**

Function **IIF** returns one of the two values depending on the value of a logical expression. The syntax is: IIF(Logical_expr, Value1, Value2)

| Invoice template | Explanation |
|---|---|
| \scan(dtLinkJobs)\ <br> ... <br> \scanentry\ <br> ... <br> \IIF(DTLINKJOBS:CJOB_ISCOMPLETED=true,100,0)\ <br> ... <br> \scanfooter\ <br> ... <br> \endscan\ | If the Client Job is completed( DTLINKJOBS:CJOB_ISCOMPLETED=true) then **100.00** is displayed in a produced invoice. If the Client Job is not completed(DTLINKJOBS:CJOB_ISCOMPLETED=false), then **0.00** is displayed in a produced invoice. |

**Numeric report functions.**

**ROUND -** The **Round** function rounds a real-type value to an integer-type value. 0.5 is always processed to largest integer number. This is not a banker's rounding.

| Invoice template | Explanation |
|---|---|
| \Round(JOBS_TOTAL)\ | If Jobs Total is 504.49, then **504** is displayed in a produced invoice, <br> If Jobs Total is 504.50, then **505** is displayed in a produced invoice. |

**INT** - The **INT** function returns the integer part of a real number.

| Invoice template | Explanation |
|---|---|
| \Int(JOBS_TOTAL)\ | If Jobs Total is 504.49, then **504** is displayed in a produced invoice, <br> If Jobs Total is 504.51, then **504** is displayed in a produced invoice. |

**SUM function**

Function **SUM** can be used after \scan(dtLinkJobs)\, \scan(dtLinkInvoices)\ or \scan(dtLinkPayments)\ to give to some new custom variable the value of sum of the values in the defined field. The syntax is:

\scan(table1)\
\endscan, sum(field of the table1, variable1)\
**Total:** \variable1\

| Invoice template | Explanation |
|---|---|

| | |
|---|---|
| \scan(dtLinkJobs)\ \endscan, sum(DTLINKJOBS:CJOB_TOTAL, V1)\ **Total:** \V1\ | Variable V1 is set to return the sum of client job totals anywhere in this invoice, just by entering \V1\ anywhere below in this invoice. If there are two client jobs in this invoice with totals of 345.00 and 678.00, then **Total:** 1023.00 will be displayed in a produced invoice. |

**CTN function**

Function **CTN** can be used after \scan(dtLinkJobs)\, \scan(dtLinkInvoices)\ or \scan(dtLinkPayments)\ to give to some new custom variable the value of number of data field entries with values <> 0. The syntax is:

\scan(table1)\
\endscan, ctn(field of the table1, variable1)\
**Total:** \variable1\

| Invoice template | Explanation |
|---|---|
| \scan(dtLinkJobs)\ \endscan, ctn(DTLINKJOBS:CJOB_TOTAL, V1)\ **Number of Client Jobs:** \V1\ | Variable V1 is set to return the number of client jobs with totals that are <> 0 anywhere in this invoice, just by entering \V1\ anywhere below in this invoice. If there are three client jobs in this invoice with totals of 345.00, 678.00 and 901.00, then **Number of Client Jobs:** 3.00 will be displayed in a produced invoice. |

**NORESET option with SUM and CTN functions**

**NORESET** option can be used with **SUM** and **CTN** functions to add the new values of the source field to the previous result of the function. The syntax is:

\scan(table1)\

........

\endscan, sum(field of the table1, variable1)\

\scan(table2)\
........

\endscan, sum(field of the table2, variable1,noreset)\

All totals: \variable1\

| Invoice template | Explanation |
|---|---|

| | |
|---|---|
| \scan(dtLinkJobs)\ \endscan, sum(DTLINKJOBS:CJOB_TOTAL, V1)\ \scan(DTLINKPAYMENTS)\ \endscan, sum(DTLINKPAYMENTS: CPAYM_TOTAL, V1,noreset)\ **Total:** \V1\ | Variable V1 is set to return the sum of client job totals plus payment totals anywhere in this invoice, just by entering \V1\ anywhere below in this invoice. If there are two client jobs in this invoice with totals of 345.00 and 678.00 and one payment with total of 77.00 , then **Total:** 1100.00 will be displayed in a produced invoice. |