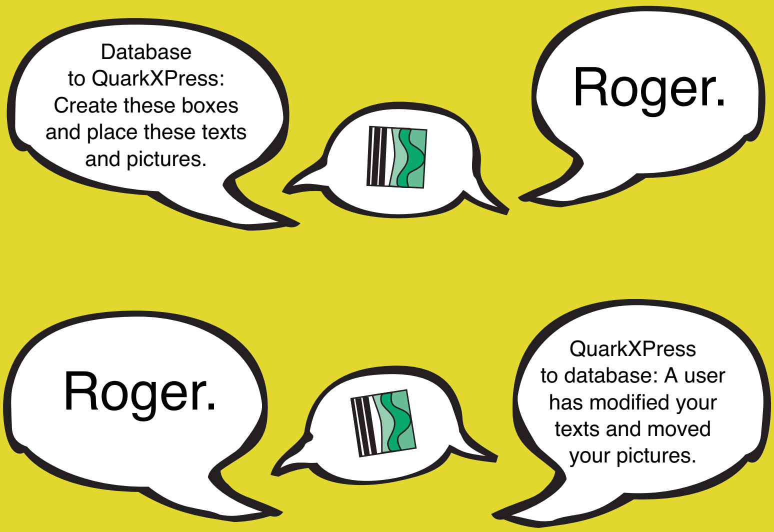
GASSENHUBER
Systementwicklung
www.gassenhuber.de

Developer handbook DATAform

XTension and DATAformTags


Database to QuarkXPress: Create these boxes and place these texts and pictures.
Roger.
Roger.
QuarkXPress to database: A user has modified your texts and moved your pictures.

Database-publishing with QuarkXPress

DATAform®
Database publishing

As of May 2007

DATAformXTension Version 5 to 7.2.1
for QuarkXPress 5 to 7.2

For MacOS, MacOS X and Windows

English translation of this manual for the most part by Renate Wismath and David Morley

DATAformXTension and DATAformMarken
are products of

GASSENHUBER Systementwicklung
Wassergasse 10
D 93059 Regensburg
Germany

www.gassenhuber.de

Visit our homepage, get news and updates!

Content

I. DATAformXTension

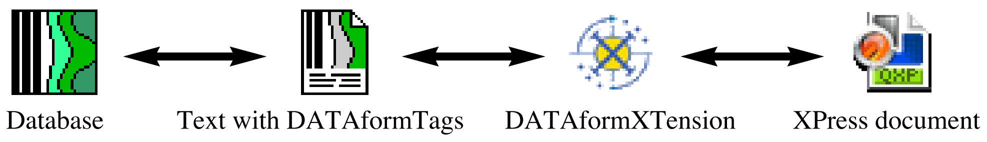
Overview – How it works

DATAformXTension is an XTension for QuarkXPress, that enables DATAformTags to be applied in QuarkXPress documents or alternatively allows you to translate QuarkXPress documents into DATAformTags.

DATAformTags form an ASCII description language to generate QuarkXPress documents, and allow QuarkXPress to be controlled from a database. DATAformXTension allows the creation of a database publishing system.

The Database generates a text with DATAformTags and sends it to the layout programme. The XTension interprets the text and creates new pages, new boxes, places pictures, lines and sets the fonts and box properties. The generated boxes and contents are free to be edited in QuarkXPress.

In order to update, XTension can translate a QuarkXPress document or parts thereof back into a text with DATAformTags. The text is then interpreted by the database and updates the data.



DATAformXTension is database neutral. It interprets an ASCII text regardless of its origin and exports an open ASCII text. It runs with QuarkXPress under MacOS and Windows operating systems; it can convert the various character sets of the operating systems.

Compatible databases include:

- DATAform database: A complete database publishing solution based on 4th Dimension, which covers many situations and demands.
- Another 4th Dimension application: Every 4th Dimension application can be extended with the DATAform developer kit to allow database publishing functions.
- Any Database in any computer environment that is capable of dealing with a complex ASCII text.

Overview – Functions

General functions

- Creating new documents and pages in QuarkXPress.
- Applying master pages on new pages.
- Creating text boxes, picture boxes, polygons and lines.
- Creating QuarkXPress- and CoolBlends-colour blends
- Marking all boxes through one object and one group number.
- Semi-automatic assignment of box ID numbers for importing “handmade” documents
- Importing from the clipboard, from a file in the QuarkXPress folder, a file in the document folder or via the “open file” dialogue.
- Updating the contents of existing text and picture boxes.
- Grouping the imported article modules.
- Background control of QuarkXPress documents via an internal message interface.
- Mutual conversion of Mac and ASCII character sets.
- Export of a complete QuarkXPress document or a selection of boxes into a database compatible text with DATAformTags.

Import and export of text boxes

- Import and export of text boxes with text, with or without XPressTags.
- Import and export of text boxes with over 70 properties (such as size, colour, frame style, flipping, trapping, box shape etc.)
- Automatic adjustment of the box height in relation to the quantity of text.
- Creating chained text boxes
- Anchoring of text and picture boxes within texts.

Import and export of lines

- Import and export of all line properties corresponding to the line dialogue, the trapping palette and the runaround dialogue.

Import and export of picture boxes

- Placement of pictures, with the help of a picture path, or part of a path, export of the picture.
- Automatic scaling of the pictures in the boxes.
- Import and export of picture boxes with over 70 properties

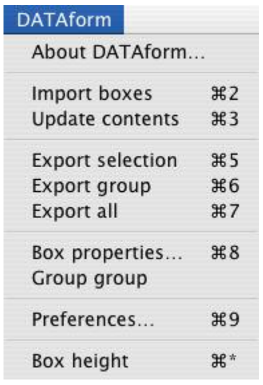
Independence of the generated QuarkXPress-Documents

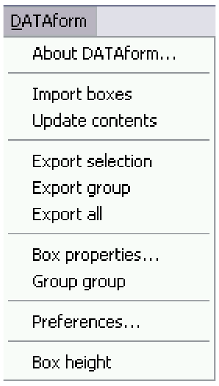
- QuarkXPress boxes or documents generated or transformed with DATAformXTension, remain independent from DATAformXTension for further editing
- Boxes that are generated or transformed by XTension, can be further edited like other QuarkXPress boxes.
- XTension edited documents can, without limitation, be further edited, printed or exposed on QuarkXPress desktops without DATAformXTension.
- Pictures placed with DATAform can be printed and used just like other pictures placed manually.

Installation and licensing

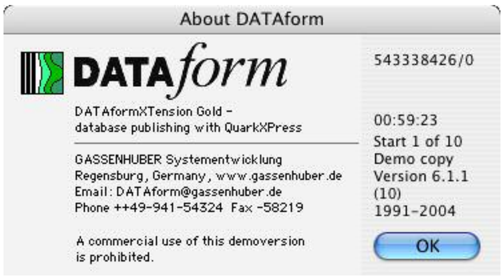
DATAformXTension installation

- Quit QuarkXPress and place DATAformXTension in the XTension folder within the QuarkXPress folder.
- Start QuarkXPress. QuarkXPress now has a new menu called DATAform:





If you wish to install the demo version of DATAformXTension, installation is already completed. Choose the command “About DATAform...” from the DATAform menu and the following dialogue will appear:



About DATAform

DATAformXTension Gold -
database publishing with QuarkXPress

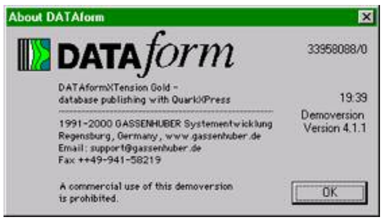
GASSENHUBER Systementwicklung
Regensburg, Germany, www.gassenhuber.de
Email: DATAform@gassenhuber.de
Phone ++49-941-54324 Fax -58219

A commercial use of this demoversion
is prohibited.

543338426/0

00:59:23
Start 1 of 10
Demo copy
Version 6.1.1
(10)
1991-2004

OK



About DATAform

DATAformXTension Gold -
database publishing with QuarkXPress

1991-2000 GASSENHUBER Systementwicklung
Regensburg, Germany, www.gassenhuber.de
Email: support@gassenhuber.de
Fax ++49-941-58219

A commercial use of this demoversion
is prohibited.

33958088/0

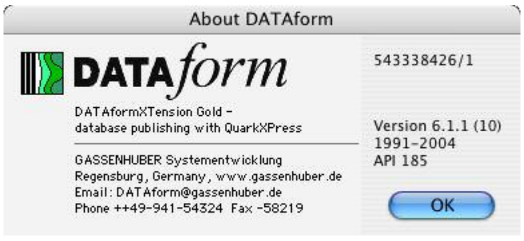
19:39
Demoversion
Version 4.1.1

OK

You can work with DATAformXTension 6 for one hour without limitation after the next 10 QuarkXPress starts (left picture). Older versions work 20 times for 20 minutes (right picture).

If you wish to install the full version, complete the following steps:

- Quit QuarkXPress once again.
  In your QuarkXPress folder you will find a folder called “DATAform”.
- Place the file “DATAform.LIZ” in this folder. The file “DATAform.LIZ” is a plain text file containing a licence code, such as:
  96732402 9184563628 8153573613
  If you received your code by e-mail paste it into a text file and save it as “DATAform.LIZ”.
- Start QuarkXPress again and open from the DATAform menu the option “About DATAform”. If you see the following smaller dialogue, DATAformXTension has been successfully installed.



About DATAform

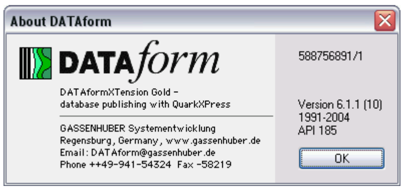
DATAformXTension Gold -
database publishing with QuarkXPress

GASSENHUBER Systementwicklung
Regensburg, Germany, www.gassenhuber.de
Email: DATAform@gassenhuber.de
Phone ++49-941-54324 Fax -58219

543338426/1

Version 6.1.1 (10)
1991-2004
API 185

OK



About DATAform

DATAformXTension Gold -
database publishing with QuarkXPress

GASSENHUBER Systementwicklung
Regensburg, Germany, www.gassenhuber.de
Email: DATAform@gassenhuber.de
Phone ++49-941-54324 Fax -58219

588756891/1

Version 6.1.1 (10)
1991-2004
API 185

OK

Multi-user licenses

If you are using more than one QuarkXPress version with DATAformXTensions in a network, please take note of the following:

- The file DATAform.LIZ contains a list of all the QuarkXPress numbers in the network that can work with DATAformXTension. All desktops that are registered in the DATAform.LIZ file must have the same DATAform.LIZ file in the QuarkXPress/DATAform folder.
- QuarkXPress desktops in a network that shouldn't work with DATAformXTension and are not registered in the DATAform.LIZ file can not install DATAformXTension. Erase XTension from these desktops.
- If you use a QuarkXPress 6 multi-user license (one QuarkXPress serial number including several licenses) you need the same amount of DATAformXTension licenses.

Checking licenses

If you have license related problems with DATAformXTension (with some QuarkXPress desktops, the DATAform demo dialogue can sometimes appear) proceed as follows:

- Open the file DATAform.LIZ by dragging it onto SimpleText if working with MacOS (under Windows rename the file temporarily as DATAform.txt and open it in Notepad.EXE). The file contains a list such as:

  ```
  96732402     9184563628      8153573613
  96732403     7384563678      9153573634
  ```

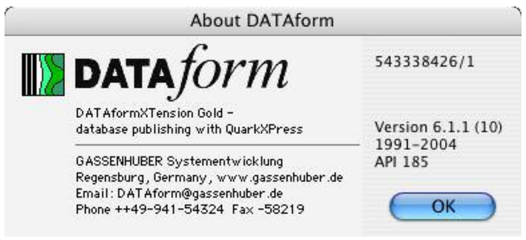
  Each line contains three large numbers that form a license for a QuarkXPress serial number (which can also be a multi-user license).
  (Under windows all numbers may appear in one line; you can insert line breaks after every third number or add new lines, see below.)

  In both cases the first number is the serial number for the QuarkXPress version for which the license is valid. In the above example the file contains licenses for the two QuarkXPress versions with the license numbers 96732402 and 96732403. The two remaining numbers are the unlock code for the serial number.

  The file DATAform.LIZ is a complete list of all the QuarkXPress versions in the network that can work with DATAformXTension.
- Compare these numbers with all the QuarkXPress versions in the network, that have DATAformXTension installed.
- Check, whether all the DATAform QuarkXPress desktops have the same DATAform.LIZ file installed.

Reading the QuarkXPress serial number

The "About DATAformXTension" dialogue shows the serial number of the QuarkXPress programme in the top right hand corner. The last digits of this number can vary from the number shown in the "About QuarkXPress" window.


About DATAform

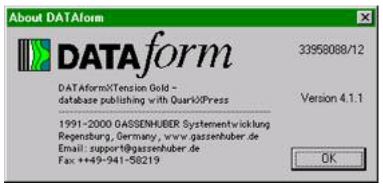
DATAform

DATAformXTension Gold –
database publishing with QuarkXPress

GASSENHUBER Systementwicklung
Regensburg, Germany, www.gassenhuber.de
Email: DATAform@gassenhuber.de
Phone ++49-941-54324 Fax -58219

543338426/1

Version 6.1.1 (10)
1991-2004
API 185


About DATAform

DATAform

DATAformXTension Gold –
database publishing with QuarkXPress

1991-2000 GASSENHUBER Systementwicklung
Regensburg, Germany, www.gassenhuber.de
Email: support@gassenhuber.de
Fax ++49-941-58219

33958088/12

Version 4.1.1

The serial number that appears in this dialogue window is the number which is vital for the licensing of DATAformXTension for this version of QuarkXPress.

After the slash you can see the number of DATAform licenses for the QuarkXPress serial number. The left hand picture shows a MacOS X QuarkXPress which has 1 DATAform license; the Windows version on the right hand side has 12 licenses.

With multi-user QuarkXPress licenses all the desktops must show the same number of licenses, otherwise the DATAform.LIZ files must be updated.

Adding licenses

- Single user QuarkXPress licenses can have their own DATAform.LIZ file. If a new QuarkXPress single user license with DATAformXTension is added, no changes need to be made on the remaining desktops.
- Multi-user QuarkXPress license
  You receive the three new digits like “96732402 9184563628 8153573613”, for example, by e-mail.
  Then open an existing old DATAform.LIZ file as described above and replace the old digits by the new one.
  Copy the new DATAform.LIZ file to all the other QuarkXPress DATAform desktops and replace the old DATAform.LIZ file.

  All multi-user QuarkXPress licenses must have the same DATAform.LIZ file. If a DATAformXTension is added to a QuarkXPress multi user license, the desktops that work with that QuarkXPress version need the new DATAform.LIZ file.

  With QuarkXPress 5 multi user licenses, the following message can be displayed:
  “At the moment all DATAform licences are in use for this QuarkXPress licence: 8585858”

  There are no DATAform licenses free for serial number 8585858 at this time. The DATAform.LIZ file contains fewer licenses than DATAformXTensions on the network are actually opened. In the dialogue “About DATAform...” you can see the number of DATAform licenses at this desktop.

DATAformTags - Quick start

DATAformTags can be written as a text, copied and transferred to QuarkXPress. (However, they would normally be generated automatically by a database.)

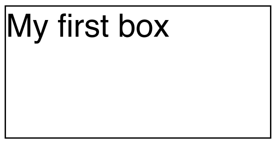
On the following three pages, DATAformTags are, as an exercise, typed into QuarkXPress and then applied. In the PDF version of this manual, you can copy the DATAformTags directly from the following lines.
This will demonstrate the basic functionality of DATAformTags. Perform the following steps on your computer:

1) Creating a new text box.

- Choose the command “Preferences...” from the DATAform menu. From here, choose the commands “Import from clipboard” and “Export via DATAform.AKT”, OK.
- Copy the following line into the clipboard:

  ¶*#1*T3*x100*X200*y100*Y150*$My first box¶
- Then choose the command “Import boxes”” from the DATAform menu.

A new box with the text “My first box” will be created on the current page.



The box will have the following coordinates:

*x100	left offest 100 pt
*y100	top offest 100 pt
*X200	right offest 200 pt, makes a width of 100 pt
*Y150	bottom offest 150 pt, makes a box height of 50 pt

Check the coordinates of the generated box in the QuarkXPress dialogue “Modify...” or in the measurements palette.

To make comparison easier, set the measuring system of QuarkXPress in QuarkXPress/Preferences/Measurements to points. Coordinates are always shown as points in DATAform.

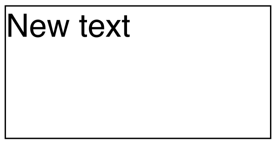
2) Updating the contents of an existing text box

- Move the previously created text box anywhere in the document.
- Copy the following line:

  ¶*#1*T3*$New text¶

- Choose the DATAform command “Update contents” from QuarkXPress.

The text in the box will be replaced by the new text:



New text

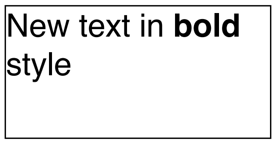
All other manually modified box properties, for example box position, remain unchanged. The box will be found in the QuarkXPress document, due to it's ID-number *#1, and updated.

- Copy the following line:

  ¶*#1*F2*T3*$New text in <PB>bold<B> style¶

- Choose the DATAform command “Update contents” from QuarkXPress once again.

The text in the box will be replaced by the new text:



New text in **bold** style

The word “bold” will be displayed as semi-bold. The box text after *$ can contain XPressTags, that define the various style properties of the text.

3) Exporting QuarkXPress objects as text.

- Draw a new text box, write “Test” in the box and give the box any properties:

- Mark the box and choose the option “Export selection” from the DATAform menu. DATAformXTension then creates a text file with the name DATAform.AKT in the Quark-XPress DATAform folder (or it opens the create file dialogue when this option is activated in the “Preferences...”.)
- Draw another text box, put the cursor in the box and choose the QuarkXPress command "Get text..." Make sure, the import dialogue options “Convert Quotes” and “Include Style Sheets” are off:

- Import the file "DATAform.AKT".

A DATAformTags description of the QuarkXPress document and the text box appears, such as:

```
DfXT+2.0¶*T107*e["FEUER:em/DF:DF Kit:DATAformXTensionE";841.89;595.276;
28.346;28.346;28.346;28.346;0;1;22.677;0]*$¶*#0*G0*T3*@1*x0*
y0*X83*Y52*W0*S0*C2*c12.024*i0*I0*j0*J0*f0*H1*h1*D0*d0*B"Black"*b0.5*F1*
1"Helvetica"*20*312*41*5"Black"*61*70*80*90*L129*l4*M["Black";"White"]*m[1;1]*
R0*r1*w0*s0*A0*a0*|0*»0*«0*§0*N0*-0*g"222222"*$Test¶
```

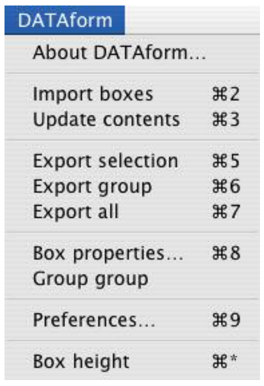
- Copy the complete text and choose the option "Import boxes". The text with DATAform-Tags will create the same box again.

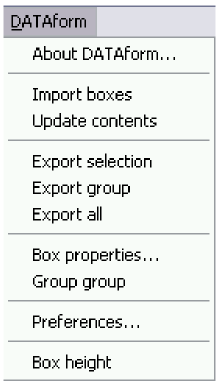
By using this method - and with the help of this manual - it is easy to find out into which DATAform parameters the properties of QuarkXPress boxes are translated.
The exported text with DATAformTags can also include XPressTags, if this option is activated in the “Preferences...”.

By this, the three basic functions DATAform have been illustrated:
- Importing text with DATAformTags in order to create new boxes and pages.
- Importing text with DATAformTags in order to update existing QuarkXPress objects.
- Exporting QuarkXPress objects as database readable ASCII text, in order to update the database.

DATAform Menu

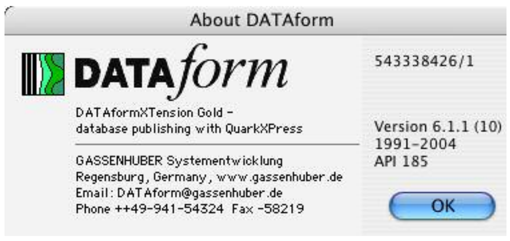
If DATAformXTension is installed, an extra menu is added to QuarkXPress:





DATAform/About DATAform...

This command opens the dialogue:


About DATAform

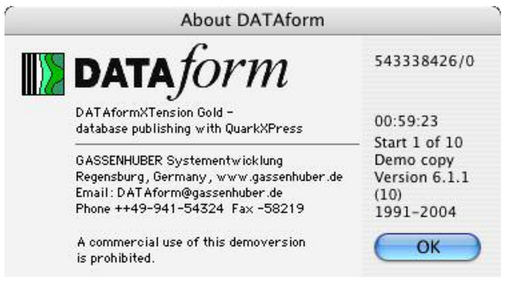
DATAformXTension Gold –
database publishing with QuarkXPress

GASSENHUBER Systementwicklung
Regensburg, Germany, www.gassenhuber.de
Email: DATAform@gassenhuber.de
Phone ++49-941-54324 Fax -58219

543338426/1

Version 6.1.1 (10)
1991-2004
API 185

OK


About DATAform

DATAformXTension Gold –
database publishing with QuarkXPress

GASSENHUBER Systementwicklung
Regensburg, Germany, www.gassenhuber.de
Email: DATAform@gassenhuber.de
Phone ++49-941-54324 Fax -58219

A commercial use of this demoversion
is prohibited.

543338426/0

00:59:23
Start 1 of 10
Demo copy
Version 6.1.1
(10)
1991-2004

OK

The left hand dialogue appears when DATAformXTension has been correctly serialised for the active QuarkXPress. To the top right you can see the QuarkXPress serial number and after the slash the number of DATAform licenses for this QuarkXPress package.

If DATAformXTension is loaded with a QuarkXPress version for which it has not been serialised, the right dialogue appears: DATAformXTension then runs a limited time only with all its functions enabled.

The instructions for installing the full version of DATAformXTension may be found in the installation chapter at the beginning of this manual, page: 8.

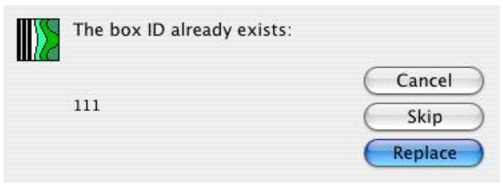
DATAform/Import boxes

This command reads a text with DATAformTags and creates the appropriate objects. Text boxes, picture boxes or lines can be imported. The text is loaded from the clipboard or from a file, as set in the preferences dialogue. (See “Preferences...”, page: 23.)

Checking for uniqueness
If an object description contains an object ID (shown by the Tag *#), then the QuarkXPress document is checked to see if a box with this ID number already exists.
If a box exists with this box ID number, then a dialogue is shown and the box can be replaced or overwritten.
Copy the following text

¶*#111*T3*x100*X200*y100*Y150*$My first box¶

and create the box.
Create the box again with the command “Import box”, the following dialogue will appear:


The box ID already exists:

111

Box N°. 111 just created by *#111 already exists.

Cancel
The complete import procedure is cancelled.

Skip
The box is not imported; the procedure skips this object and proceeds with the next object.

Replace
The existing box is deleted and the new box will be created.
The new box is created at the position and the page, as defined by its Tags. Its properties are independent of the properties of the previously deleted box.

If the object number is *#0 or if the Tag is missing, there is no check to see whether the box already exists, the object will always be created.

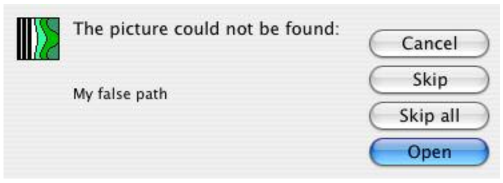
Import picture boxes and loading pictures
If the text with DATAformTags describes a picture box with a path to a picture, then DATAformXTension will search for the picture and place it in the box. If the supplied path is a complete picture path, it looks for the picture at this location then in the document folder and finally in the QuarkXPress folder. (See under DATAformTags *{1} *$ Text/path*, page: 32)
If the picture is found, QuarkXPress will load it as if it had been placed manually. DATAformXTension passes the path to the picture after checking it onto QuarkXPress; Quark-XPress itself then loads the picture. All QuarkXPress picture formats or OPI functions are available.

If the picture is not found, you have the possibility to load it manually.

¶*#2*T13*x100*X200*y100*Y150*$My false path¶

If you import this text with DATAformTags, DATAform expects, due to *T13, a picture path after *$.

If the picture “My false path” cannot be found, the following dialogue will appear:


The picture could not be found:

My false path

<u>Cancel</u>
The picture box is created, the picture is not loaded.
The complete import procedure is cancelled.

<u>Skip</u>
The box is created, the picture is not loaded. The procedure continues with the next object.

<u>Skip all</u>
Clicking this option suppresses the dialogue for this import session. If further pictures cannot be found, their boxes will be created, but the loading process of the picture will be skipped.

<u>Open</u>
The QuarkXPress load picture dialogue is opened, and a picture can be chosen and placed. If you click on Cancel in this dialogue, the picture will be skipped and DATAform continues with the next object.

Functional objects
With functional objects you can suppress missing picture alerts. This may be desired, particularly when working with the message interface, see page: 93.

Import picture box
If you specify an empty picture path, only the picture box will be placed and the picture will not be looked for:

¶*#2*T13*x100*X200*y100*Y150*$¶

Creates an empty picture box, cf. *{1} *$ Text/path,* page: 32.

Positioning, master pages
The imported objects will be placed either on the current QuarkXPress page or on the page, which is marked with *{6} *p Page*. (See the description of *p, page: 38)

Ojects cannot be placed on master pages. Master pages can however be applied to new pages (See under *{3} *P Master page,* page: 35)

DATAform/Update contents

This command loads a text with DATAformTags, looks for the boxes in the active Quark-XPress document and updates their contents.
Only boxes with a box ID number other than zero can be updated. A box ID can be assigned either when being imported via the *# Tag or manually by using the command “Box properties...” (See under “Box properties...”, page: 21.)

Objects on master pages cannot be updated.

Text or picture boxes that are anchored in a text box can be updated and remain anchored.

The text in chained boxes can be updated as a whole.

If the object is found, its content will be replaced by the new content, i.e. the text will be overwritten, a new picture will be placed. The command “Update contents...” performs different functions on the three object types, text boxes, picture boxes and lines.

Update Text boxes

In text boxes, “Update contents...” overwrites the current text.

Box properties
The properties of a text box such as frame style, background etc., remain unchanged. The position of the top or bottom edge of a text box, can be changed by the Tag *{60} *Δ Adjust box* height: If the quantity of text changes, the box will automatically adjust to the new amount of text. (See under *{60}*Δ Adjust box height, Page: 74*)

Font mode
There are three ways to set the font and style of the text by the DATAformTag *F:

*F0	The box font will be left unchanged.
*F1	The DATAformTags for the box font properties *1 to *9 are used, the box font is set accordingly.
*F2	The XPressTags contained in the object description are applied.

(See under DATAformTags *{28} *F Font mode*, page: 51.)

Chained boxes
If text boxes are chained, the entire chained text is replaced with the new text when updated. The text is therefore transferred with the chains first box. (See under *{64},{65}*», *« Chained boxes*, page: 77.)

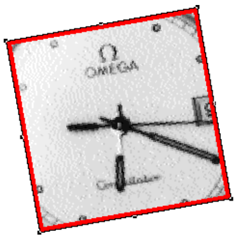
Update Picture boxes

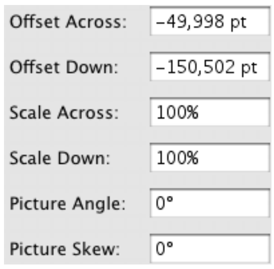
With picture frames the new picture is loaded when updating. The properties of the picture, such as position, angle and background colour remain unchanged. Likewise the picture properties such as scaling and offset remain untouched. When updating a picture the new picture is placed with the same properties as the old one.

Example:

¶*W10*L128*l2*M"Red"*m1*k1*K1*Z1*#8*T12*$:Omega.tif¶

DATAform/Import boxes, when used with the above line, creates a picture box with the N°. #8 loads the picture from the QuarkXPress documents folder and centres it. The picture box is turned 10 degrees and framed with a 2pt line, see left picture:



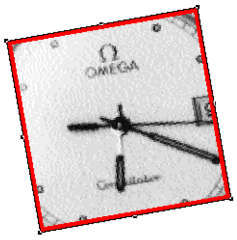


Offset Across:	–49,998 pt
Offset Down:	–150,502 pt
Scale Across:	100%
Scale Down:	100%
Picture Angle:	0°
Picture Skew:	0°

The right screenshot shows the offset, which is calculated and set by *Z1 (centered picture)

¶*W0*L128*l2*M"Blue"*m1*k1*K1*Z3*#8*T12*$:Omega.tif¶

DATAform/Update contents looks for box #8 and loads the picture once again:



The current picture position remains unchanged.

All other picture and box properties remain unchanged: Hence the above tags *W0*L128*l2*M"Blue"*m1 (box angle 0 degrees, blue line etc.) are not applied.

Update Lines
Colour, shade, width, style and arrowheads are newly set.
Position, angle, and length remain unchanged.

¶*#3*T0*l5*E1*m1*M"Red"*$¶

“Import boxes” using this line creates a red arrow.

¶*#3*m1*M"Black"*T0*l5*E5*$¶

“Update contents” using this line changes the red arrow into a black double arrow at the same position.

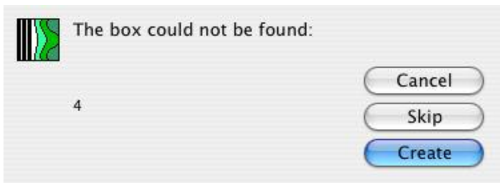
Unfound boxes

If an object is not found during the update process, it can be skipped or replaced via a dialogue box.

Example: Copy the following text

¶*T3*#4*$My new box¶

and choose the command DATAform/Update contents.
If there are no boxes with ID 4 in the QuarkXPress document, the following dialogue appears:


The box could not be found:

4

<u>Cancel</u>
The complete updating process is cancelled.

<u>Skip</u>
The box is not updated; the procedure skips the object and proceeds with the next one.

<u>Create</u>
The box that could not be found is newly created.

Unfound pictures when updating picture boxes

If a picture cannot be found during the update procedure, the DATAform load picture dialogue appears. The picture may be loaded or skipped. (See under “Import boxes”, page: 15)

DATAform/Export selection, - group, - all

These three commands are used for updating the database with the QuarkXPress document. The commands create a text file with DATAformTags with the box properties and contents. The file can then be loaded into the database and the corresponding records will be updated. Depending on the settings in “Preferences...” the location of the file will either be determined via dialogue or a file with the name DATAform.AKT will be created in the QuarkXPress folder. See under “Preferences...” in the DATAform menu.

With these three commands, all objects will be exported: the ones created by XTension, as well as those created manually in QuarkXPress. Objects on master pages are not exported.

Export selection

This command only writes the selected box or boxes in QuarkXPress into the DATAformTags text file.

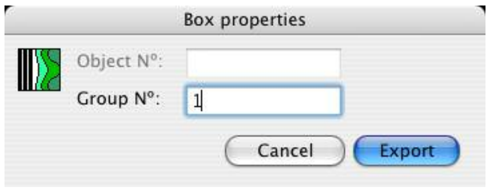
Export group

Several boxes can be grouped together by their ID numbers. For example several boxes form a group in order to present an article: text, pictures, eye catchers etc.

The command “Export group” exports all those boxes that belong to the same group (all boxes marked with the same group number in the “Box properties...” dialogue).

A box with a group number is selected
If a box with a group number other than zero or several boxes with this group number are selected, all the boxes of the document with the same group number are exported.

No boxes or different boxes are selected
If no boxes or boxes with different group numbers are selected, then a dialogue appears asking you to enter the group number:


Box properties
Object N°:
Group N°: 1

In this setting all the boxes in a QuarkXPress document with group number 1 are exported. Group numbers are assigned to a box when importing and using the Tag *{4} *G* or manually via the “Box properties” dialogue. See under “Box properties”, page: 21.

DATAform/Export all

DATAform/Export all writes all the objects of a QuarkXPress document into the text file with DATAformTags.

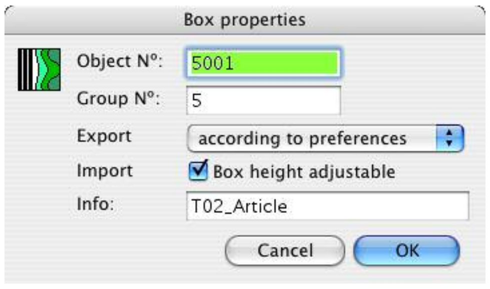
Export anchored boxes
Boxes that are anchored in a text box are, like other boxes, exported as independent boxes. The anchoring information is placed in the text of the main box between {{ }} and with the Tag *{66} *A Anchored in box*, cf. page: 79. All boxes must have unique box ID numbers before being exported.
When imported once more, the boxes will be anchored again.

Export chained boxes
Chained boxes are exported like single unchained boxes. The entire chained text will be exported with the first box. The chaining information "next box" and "previous box" will be placed with the Tags {64} *>> and {65} *<<, cf. page: 77.
When imported once more, the boxes will be chained again.

DATAform/Box properties...

Box IDs are used for the identification and bi-directional updating of QuarkXPress boxes and a database. The numbers are attached to the boxes (text boxes, picture boxes or lines) during import and are exported with the Tag *# and *G. Via the Box properties dialogue, these numbers can be read and manually changed or newly assigned. The other fields and properties of this dialogue are described at *{75} *g Box properties*, page: 89.

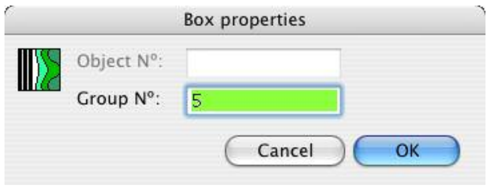
One box is selected
If only one box is selected, this command opens a dialogue to read and change both box ID numbers:


Box properties

Object N°:	5001
Group N°:	5
Export	according to preferences
Import	☑ Box height adjustable
Info:	T02_Article

Control duplicates
If you click on OK in the above shown illustration, DATAform checks that all object numbers are unique. If the number is already in use in the document, you will get an alert and have to change the object number.

Several boxes are selected
If several boxes or groups of boxes are selected, the group number for all active boxes can be seen and modified:


Box properties

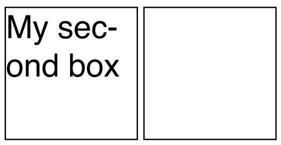
Object N°:	
Group N°:	5

If the active boxes have the same group number, it will be displayed in the dialogue. If the boxes have different group numbers, no number will be shown. In both cases, the group number of all active boxes can be overwritten in one step via the dialogue. The selection of boxes may also include grouped boxes.

Assign Box IDs when importing
When importing, boxes receive an object number with the Tag *#, and a group number with the Tag *G. The text:

¶*#5*G1*T3*x300*X350*y100*Y150*$My second box¶*#6*G1*T3*x350*X400*y100*Y150*$¶

creates two boxes, both of which belong to group 1.



The first box has object number 5, the second has object number 6.

*Object N°.: *#*
The object number serves as a way of clearly identifying a box within a QuarkXPress document. A database should always assign unique object numbers. When exporting QuarkXPress boxes, it gets these numbers back so that the corresponding database records can be found and updated.

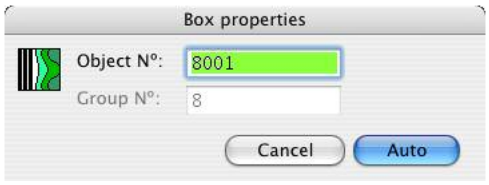
DATAformXTension checks when importing objects that they are unique.
If a Tag *# other than zero is placed when importing XTension checks the document for boxes with this object number. If an object with this number is found, a message appears. (See under “Import boxes”, page: 15)
Boxes with the same object numbers cannot be imported.

*Group N°.: *G*
The group number allows the grouping of several boxes . The number is important in three ways:

- The command “Export group” exports all boxes in a document with a specific group number. (See under “Export group”, page: 20.)
- The command “Group group” puts all boxes with the same group number into a QuarkXPress group. (See under “Group group”, page: 23.)
- If in the preferences dialogue “Group” is activated, boxes with the same group number will be grouped during import. (See under “Preferences...”, page: 24.)

Advanced Dialogue Properties
The fields of the dialogue can be deactivated or hidden per box. This way box properties can be individualized or concealed from the user, see *{75} *g Box properties*, page: 89.

Automated box numbering
In order to import “hand made” QuarkXPress documents boxes must be numbered with object and group numbers module by module. This operation is facilitated by the automated box numbering of DATAformXTensions. It automatically numbers all boxes in an article’s module. To achieve this you group all boxes of the module and mark the main text box with the QuarkXPress content tools. If one calls the Box properties dialogue in this situation, the following dialogue appears:



Box properties

Object N°: 8001

Group N°: 8

DATAformXTension proposes the next highest available object number. If you click on “Auto”, all boxes in the module receive a consecutive object number, and in the above example, the group number 8. The entire procedure is described in detail in the chapter “Importing hand made documents”, page: 137.

DATAform/Group group

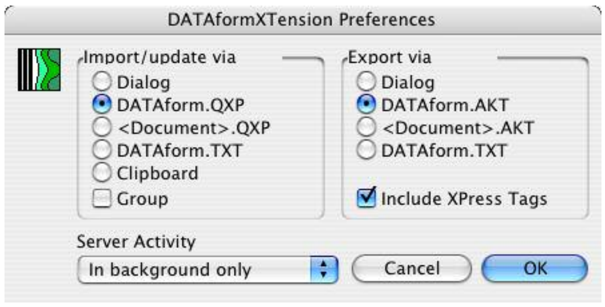
This command groups all boxes of a DATAform group into a QuarkXPress group.

You mark a DATAform box (i.e. a box with a box ID number and a group number other than zero) and call up the command. All boxes that have the same group number are grouped as a QuarkXPress group. Only the boxes on the same page or spread are grouped.
If boxes are member of other groups, they are taken out of these groups.

The command works like an import with the option “Group”. It enables you to group these boxes later or to group them again after they have been ungrouped.

DATAform/Preferences...

This command opens the dialogue:



DATAformXTension Preferences

Import/update via
- Dialog
- [✓ DATAform.QXP]
- <Document>.QXP
- DATAform.TXT
- Clipboard
- ☐ Group

Export via
- Dialog
- [✓ DATAform.AKT]
- <Document>.AKT
- DATAform.TXT
- ✓ Include XPress Tags

Server Activity: In background only

The dialogue consists of 3 areas:
1) Import
2) Export
3) Server activity

1) Import

Dialogue
Under this configuration both import commands “Import boxes” and “Update contents” open the system dialogue enabling you to choose a text file. If you open a DATAformTags file, the boxes will be imported or the contents of existing boxes will be updated.

DATAform.QXP
Under this configuration both import commands look for a file named “DATAform.QXP” in the QuarkXPress/DATAform folder and import it. A database can place its transfer file with this name in the QuarkXPress/DATAform folder and it can be imported directly by DATAformXTension without using an open dialogue. This is the default setting.

<Document>.QXP
Under this configuration, both import commands look for a file in the active QuarkXPress documents folder called “Document_Name.QXP”. The update files can be placed alongside the QuarkXPress documents and used at will.
This configuration gives the possibility to generate any number of document related update files that may be processed at any given time.

DATAform.TXT
Under this configuration the import commands look for a file named “DATAform.TXT” in the QuarkXPress/DATAform folder and import it. The settings allows you to work with the same file on export as on import. Exported boxes can be re-imported directly. The DATAform tags file can be used as a kind of clipboard, mainly for developing purposes.

Clipboard
Under this configuration both import commands “Import boxes” and “Update contents” read the text with DATAformTags from the clipboard and apply it onto the active QuarkXPress document.

Grouping
This function allows the automatic grouping of module boxes upon import. The grouping is the normal QuarkXPress grouping and can also be undone in QuarkXPress. After importing, the module can be easily identified and shifted in the QuarkXPress page.
If the checkbox is ticked, all boxes that have the same group number will be integrated as one group as long as they are placed on the same page or spread. If boxes with the same group numbers are created on different pages, they will be grouped in separate groups.
Only DATAform boxes will be grouped; boxes with the group number zero are not grouped.

2) Export

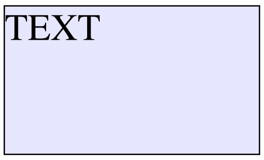
Dialogue
Under this configuration the three export functions open the system dialogue to create a new file. The name and location of the text with DATAformTags can be defined from this dialogue.

DATAform.AKT
Under this configuration the three export functions create a file named DATAform.AKT in the QuarkXPress/DATAform folder. If the QuarkXPress folder is known by the database, this setting allows the updating of the database without using the open file dialogue. This is the default configuration.

<Dokument>.AKT
Under this configuration the three export functions create a file named “Name_of_the_Document.AKT” alongside the QuarkXPress document. The option is corresponding to <Document>.QXP: in that case the import commands look for a file in the active QuarkXPress documents folder called “Name_of_the_Document.QXP”.

DATAform.TXT
Under this configuration the export commands write into the file named “DATAform.TXT” in the QuarkXPress/DATAform folder. The settings allows you to work with the same file on export as on import. Exported boxes can be re-imported directly.

Include XPressTags
If this option is active when exporting, the XPressTags of the text will be written into the text. The XPressTags will be produced after the DATAformTag *$ as part of the text. As an example, a box with the word TEXT



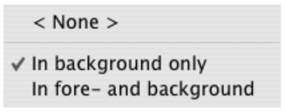
that is formatted with the style sheet “Headline”, will result in the text with DATAformTags:

```
DfXT+2.0¶*T107*e["FEUER:em/DF:DF Kit:DFXTensionHandbuch:DATAformXTen-
sionE";841.89;595.276;28.346;28.346;28.346;28.346;0;1;22.677;0]*$¶*#0*G0*T3*@1*x0*
y0*X96*Y56*W0*S0*C1*c12.024*i0*I0*j0*J0*f0*H1*h1*D0*d0*B"Blue"*b0.1*F2*
L128*l0.5*M["Black";"White"]*m[1;1]*R0*r1*w0*s0*A0*a0*|0*»0*«0*§0*N0*
-0*g"222222"*$<v6.10><e0>
@Normal=<Ps100t0h100z10k0b0cKf"Helvetica">
@Normal=[S"","Normal","Normal"]<*L*h"Standard"*kn0*kt0*ra0*rb0*d0*p(0,0,0,0,0,0,g,"
German")>
@Text=[S"","Text"]<*L*h"Standard"*kn0*kt0*ra0*rb0*d0*p(0,0,0,0,0,0,g,"German")*t(28.3
46,0,"1  "56.693,0,"1  "85.039,0,"1  "113.386,0,"1  "170.079,0,"1
")Ps100t0h100z10k0b0cKf"Times-Roman">
@Headline=[S"Text","Headline"]<*p(0,0,0,0,0,0,g,"U.S. English")*t(40,0,"1  "76,0,"1
"206,0,"1  ")Ps100t0h100z14k0b0cKf"Times-Roman">
@Headline:TEXT¶
```

The italic parts after the tag *$ are the exported XPressTags; all XPressTags including the style sheet definitions are exported. At the end of the XPressTags comes the box content “TEXT”. XPressTags can also be present within the text and change the style and font properties in paragraphs or letter by letter.

The parameter *F is set to *F2 when exporting with XPressTags. This allows a database to tell whether the text includes XPressTags or not. When exporting a text with XPressTags towards QuarkXPress the parameter *F must be set to *F2 so that the XPressTags will be interpreted again.
(See under *{1} *$ Text/Path*, page: 32, *{28} *F Font mode*, page: 51, and in appendix under “Text with XpressTags”, page: 132.)

3) Server Activity



< None >
✓ In background only
In fore- and background

With this configuration the DATAformXTension message interface is activated. A database can now send messages to QuarkXPress that will be processed in the background. More on this topic in the chapter “Messages to QuarkXPress”, page: 109.

DATAform/Box height

This command adjusts the height of a marked text box to the quantity of text present. The bottom edge of the box will be shifted so that text overflow is avoided. Depending on the quantity of text, the box will be enlarged or reduced in size. If the quantity of text is too big, the box will be stretched till the bottom edge of the page and the overflow symbol will appear.

The function has the same effect as the Tag *Δ1 when importing a text with DATAformTags.
(See under *{60} *Δ Adjust box height*, page: 74)
Via the database all placed boxes can have their height automatically adjusted by that Tag.

Various functions

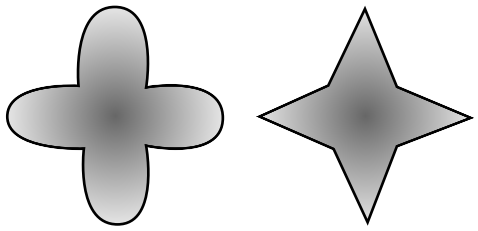
Long import or export procedures can be interrupted by means of the following key commands:

MacOS	Command-period
Windows	Escape

Limitations

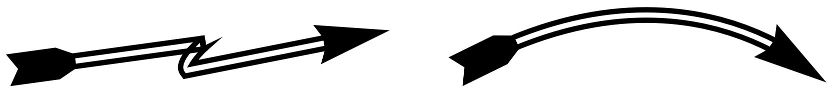
Some QuarkXPress functions are not supported by DATAformXTension. Some object properties do not remain unchanged upon export and re-import. Apart from this, these objects may be exported and imported - with all their other properties.
Not supported are e.g.:

- Bezier objects
  Bezier objects are exported as polygons. This means that the object is retained when imported with the exception of bent lines. All lines are straight.
  A form with round ends receives corners instead when imported:

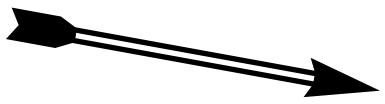


  The left picture shows an object with Bezier lines, the right picture shows the result after export and re-importing: the curves are lost.

- Non straight lines
  Polygon lines and bezier lines are exported as straight lines. All other QuarkXPress line properties are retained.



  These two lines will appear as straight lines after re-import:



- Text to path

  This QuarkXPress function is not supported;

Line with text

  A line with text is exported as just a line. The text is lost.

Workarounds
If you do need objects with Bezier lines you may draw them in a grafic program and load them in QuarkXPress as a picture. You could also generate the Bezier object directly in Quark-XPress, save it as an Eps picture or Pdf file and load the picture file.

II. DATAformTags

Creating text with DATAformTags

A text with DATAformTags can be created by any database that is capable of producing a complex ASCII text.

DATAform 4D interface

Special export and import routines for DATAformTags exist for the database 4D from 4D.inc. The procedures can be implemented in existing 4D databases and make it possible, for example, to extend an article database with database publishing functions. Installation and functionality of the interface are described in the appendix under "DATAform 4D interface", page: 121.

The DATAform 4D interface includes a translator for DATAformTags into the text array DX_Array and can likewise generate from the DX_Array a text with DATAformTags. The DATAform 4D interface user only has to write to and read from the DX_Array; he has no direct contact to the text with DATAformTags and only works with the DX_Array.

All routines of the DATAform 4D interface are available on the DATAform CD as text files in a Pascal-like language and can, starting from there, be translated into any other computer language.

Syntax rules

*1) Tag characters **
A DATAformTag begins with the character * (ASCII-42), followed by the Tag marker and the value which is assigned to the Tag. Tag markers are case sensitive.

Example 1: *p7
The Tag *p stands for the page number, on which a box will be placed. The value of the Tag is 7, i.e. the box will be placed on page 7.

Example 2: *T3
The Tag *T describes the type of box to be created. The value of the Tag is 3. A box of type *T3 is a text box, *T0 is a line for example.

2) Object delimiter ¶
Objects are text boxes, picture boxes or lines.
An object description begins and ends with the ¶ character. On both platforms, this character has the ASCII value 166. Under MacOS it is created by option-key-3, under Windows with the key combination alt+0166. Under MacOS the character appears as ¶, under windows mostly as ¦. (See under "Special characters" and "Character set tables" in the appendix.)

As an alternative to character 166 you can use character 182 as object delimiter on both platforms. Character 182 appears under Windows as ¶. (On export DATAformXTension uses character 166 always). See also point 4) and 11).

Example: ¶*#1*T3*x100*X200*y100*Y150*$My first box¶

Several descriptions are directly joined together as follows:
¶Object 1¶Object 2¶Object n¶
A text with DATAformTags can contain as many objects as desired.

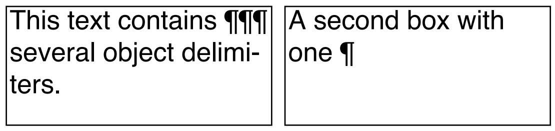
*3) *T, order, *$*
Every object description must contain the Tag *Tx for the object type. The text that will be placed in a box or the path to a picture to be placed in a picture box follows the DATAformTag *$. The text or path comes immediately after the Tag *$; it is not placed in inverted commas. XPressTags are part of the text. The Tag *$ must be included in every object description including lines and must always be the last Tag.
The order of the other DATAformTags is arbitrary.

4) Object delimiter within the text: ¶
The text in a box may contain the character ¶.
Even though the characters 166 and 182 are reserved as object delimiters – cf. point 2 – the text in a box is allowed to contain these characters. (Since DATAformXTension 4.1.8) Strictly speaking DATAformXTension takes the two successive characters ¶* as an object end (or reads to the end of file), cf. point 11.

Example:

¶*T3*$This text contains ¶¶¶ several object delimiters.¶*T3*$A second box with one ¶¶

Creates on MacOS and Windows the two boxes:


This text contains ¶¶¶ several object delimi- ters.

A second box with one ¶

5) Unauthorized characters in a text with XPressTags: < @ \
If XpressTags are applied, the characters are reserved by QuarkXPress for the beginning of XpressTags and style sheet names.
Then the text itself can not contain these characters. If the text needs these characters, they may be created by using the XPressTag <\\<>, <\\@> and <\\\\>. (See the QuarkXPress manuals.)

6) Unauthorized characters in names: @ : " ¶ { } ; =
In font names, colour names, master page names, in the prefix of the section pages in Quark-XPress as well as in style sheet names these characters can not be used.

7) Measurements , names, percentages

<u>Measurements</u>
All measurements are transferred in points.
A comma or a point can be used as a decimal seperator: *x35,7 is the same as *x35.7. Any number of decimal places can be transferred, however QuarkXPress reads these with an accu-racy of 3 decimal places only. Any extra decimal places are rounded up or down upon import. Up to 5, they are rounded down, from 6 on, they are rounded up.
When exporting boxes with DATAformXTension the accuracy of the decimal values is that of QuarkXPress.
Thousand seperators can not be used.

<u>Names</u>
Names are transferred between "inch signs" (shift+2, ASCII-34). Names are used for colours, colour blends, box fonts, section and master pages.
*B"White" sets the boxes background colour, *1"Helvetica" the boxes font, *P"MasterA" the masterpage. Both inverted commas have the same ASCII value (ASCII-34.) The character is available on the keyboard with shift+2. To create it from QuarkXPress the option

☐ Smart Quotes

must be off in the application preferences. When importing a text with DATAformTags as text into a QuarkXPress box the option

☐ Convert Quotes

must be off in the Get Text dialogue.

Names in the 4D interface
If names are transferred directly to the DX_Array line, inch signs are created with char(34).
DX_Array{25}:=char(34)+"Red"+char(34) sets the box background red.

Percentages
Percentages are given as decimal values. 100% equals 1. *b0.15 sets the shade of the box colour to 15%.

8) Validity
DATAformTags are applied in general. The Tags *x and *X for positions are used for text and picture boxes as well as for lines; Line properties are used for lines as well as for the frame style of text or picture boxes etc.

9) Missing Tags
Missing Tags are tolerated.
The following line creates two boxes, the second object description exists only of
¶*T3*$¶
and contains the minimum information required:
¶*#2*T3*x200*X300*y100*Y150*$My box¶*T3*$¶

10) Version identification
Upon export, DATAformXTension creates the header DfXT+2.0 before the first object.
The header is also set by the DATAform 4d interface.
In order to retain compatibility of archived text with DATAformTags with future versions of XTension, the DATAformTag file should always start with this header.

11) Visual organization
To enhance the optical arrangement of a DATAformTags text you may enter carriage returns after a *. But not after a ¶, because the * must succeed the ¶. Cf. point 2) and 4). Carriage returns after a * are tolerated however. Example:

```
¶*T3*x20*X120*$A box with ¶¶¶ in the text.¶*
T3*x140*X240*$A second box ¶*
T3*x260*X360*$A third box ¶
```

The text generates one box per line.

DATAformTags description

The Tags are described in the order in which they reside in the Array DX_Array in the DATAform 4D interface. The line number in the Array is marked with {Nr}.

An arrow => after the Tag description means the Tag is used upon import into QuarkXPress only and is not exported. This counts for the following Tags:
*{3} *P Master page =>*
*{51} *Z Picture position =>*
*{60} *Δ Adjust box heights =>*
*{61} *x> Offset > right =>*
*{62} *y> Offset > down =>*

An arrow <= after the Tag description means the Tag is used upon export from QuarkXPress only. This counts for the Tag: *{63} *+ Section page <=*, page: 76.

All other Tags are imported and exported.

The range of values of the Tags is placed inside the symbols *[ ]*.
[-75…75] means: only values between –75 and +75 are allowed.
[1;2] means: only values 1 or 2 are allowed.
The transfer of unauthorized values can cause QuarkXPress to crash.

{1} *$ Text/Path

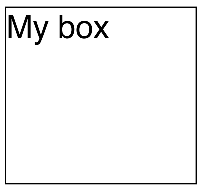
Text [unlimited]/Path [1–255 characters]

This Tag defines the end of a series of DATAformTags and the beginning of the boxes text or the picture path. The text or the path to a picture file starts immediately after the Tag marker. The Tag *$ must be present in every DATAform object description and be the last one. The shortest object is:
¶*T3*$¶
The text or path after the Tag *$ is closed with the object end character ¶. The character ¶ can not be included in the text or path itself. (see under Syntax rules 4, page: 29.)
DATAformXTension interprets the text or path depending on the box type definition, either as text for a text box, or as a picture path for a picture box. The box type is defined by the Tag {5} *T Box type. Text and picture paths are written without inverted commas.

***$ with text boxes**

If a text box is created, the text will be transferred after *$

¶*T3*$My box¶
creates a text box with the text "My box"



When exporting text boxes with DATAformXTension the boxes´ text is once again exported after the Tag *$. Depending on the "Preferences...", the text will be exported with or without XPressTags. (For more details, look under "Text with XPressTags", page: 132.)

With chained boxes, the text of the complete chain is transferred with the first box.

***$ with picture boxes**

If a picture box is created, the picture path will be transferred after *$.
To transfer a picture path into DATAformXTension there are three options:

1) Complete picture path
- If a complete picture path is transferred, it will first be looked for at this location, then
- in the QuarkXPress documents folder and finally, if it hasn't yet been found,
- in the QuarkXPress folder.

¶*T12*$HD320:PictureFolder:MyPicture¶
creates a picture box and looks for the picture "MyPicture" in the folder "PictureFolder".

2) Picture name preceded by colon
If only the picture name preceded by a colon is given as the path, the picture will be looked for first
- in the QuarkXPress documents folder and then, if not found,
- in the QuarkXPress folder.

¶*T12*$:MyPicture¶
creates a picture box and looks for the picture "MyPicture" first in the QuarkXPress documents folder and then in the QuarkXPress folder.

To look for the picture in the folder of the QuarkXPress document, the QuarkXPress document has to have been saved previously; the search will be skipped otherwise.

3) Picture name
If only a picture name is transferred, the picture is looked for in the QuarkXPress folder only.

¶*T12*$MyPicture¶
creates a picture box and looks for the file named “MyPicture” in the QuarkXPress folder.

The purpose of this feature is the option to keep pictures together with their QuarkXPress document in one folder and to place them by using :PictureName. Text with DATAformTags hence becomes independent of the filing structure of the computer with which the text with DATAformTags will be placed in QuarkXPress.

If a picture is not found, the DATAform load picture dialogue opens and a picture can be placed. (See under DATAformXTension “Import boxes”.)
Pictures are placed as if they were manually placed in QuarkXPress meaning that DATAformXTension gives QuarkXPress the picture path and lets QuarkXPress load the picture. All QuarkXPress functions related to updating and printing pictures, remain at your disposal.

When exporting picture boxes with DATAformXTension, the complete path will be exported, independent of the type of import path given after the Tag *$.
Pictures which have been placed via the clipboard (without a link to a picture file), are not exported. In this case only the picture box with an empty path after the Tag *$ is exported.

***$ with empty picture boxes**

If a picture box without path details is imported, an empty picture box is created. The picture is not looked for; The DATAform load file dialogue is not opened.

¶*T12*-0*B"Black"*b0,30*$¶
creates a 30% grey rectangle.

¶*T14*-0*B"Red"*b0,5*$¶
creates a red circle with 50% shade.

***$ with lines**

With lines no details are required after *$. Nevertheless the Tag *$ must be the last Tag for the object. The shortest text for a line is: ¶*T0*$¶

{2} *# Object N°.

[Long integer]

This Tag creates a box ID number on the placed object. In QuarkXPress, the number is saved with the box. It can be read and changed via the Box properties dialogue in the DATAform menu. Otherwise it has no influence on the administration or printing of the object from QuarkXPress.
The object number is of crucial importance for the identification of boxes in a QuarkXPress document and for the bi-directional exchange with a database. Unique numbers are crucial for the mutual updating of the database and the QuarkXPress (See under “Box properties” and “Importing hand made documents”, page: 137)

If a Tag other than zero is transferred, DATAformXTension checks upon import that the object is unique. (See under “Import boxes”, page: 15.)

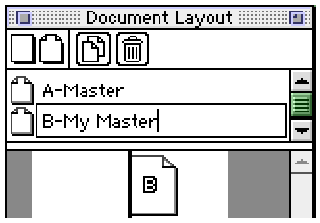
No checking is done if the object number *#0 (zero) is transferred, or the Tag is missing. If objects are duplicated in QuarkXPress, their object numbers are duplicated, too!

¶*T0*#-317*$¶
creates a line with the object number –317.

{3} *P Master page =>

["Name"]

This Tag defines which master page should be used, when a new page is created. (See more under *{6} *p Page*, page: 38.)
The name of the master page is transferred after *P.



Master page names can be changed in the “Document Layout” palette.
In QuarkXPress master page names always come after one to three ID letters, which are followed by a hyphen. The ID letters are not part of the name. The name must be enclosed in inch signs .

¶*T3*p30*P"My Master"*$Page 30¶
creates 29 new pages in a new document with the master page “My Master”. The new pages will show the ID letter “B”. A text box containing the text “Page 30” is created on page 30.

The Tag is not exported.

{4} *G Group N°.

[Long integer]

This Tag creates a group number on the placed object. The number is saved along with the box by QuarkXPress. It can be read and changed via the Box properties dialogue in the DATAform Menu. (See under DATAformXTension “Box properties…”, page: 21.)
The group number allows the integration of several boxes into one group. The number is important in three aspects:

- The command “Export group” exports all boxes of a document, that have the same group number. (See under “Export group”, page: 20.)
- The command “Group groups” integrates all boxes with the same group number into a QuarkXPress group. (See under “Group group”, page: 23.)
- If “Group” is active in the preferences dialogue, all boxes with the same group number are grouped upon import. (See under “Preferences…”, page: 24.)

In practice several boxes that are e.g. part of an article (text, picture, eye catcher), are assigned the same group number for the transfer to QuarkXPress. They can therefore be imported as a QuarkXPress group and easily be identified and shifted after placement.

¶*T3*G1*#1*$Text1¶*T3*G1*#2*$Text2¶*G1*#3*T12*$:MyPicture¶

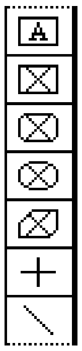
creates three boxes all of which have the group number 1.

{5} *T Object type

[0;1;3…5;12…14]

DATAformXTension contains
- box objects *T0 up to *T13 i.e. boxes and lines in QuarkXPress and
- functional objects from *T101 on which control the mode of import etc. By *103, for example, a box can be deleted; *108 creates a new document. Functional objects are described in chapter III, from page: 93.

A box object describes the type of QuarkXPress box.



*T3, *T4, *T5	Text box, *T3 square, *T4 box shape, *T5 oval
*T12	Orthogonal picture box
*T13	Various picture boxes
*T14	Oval picture boxes
*T3, *T12	Polygon as text or picture box with *t [...]
*T1	Orthogonal line
*T0	Free line

The illustration shows the available box types; any other values for *T can not be used in this context.

If the command “Update boxes” is applied to a text with DATAformTags, the type of box specified in *T must match the type of box that is being updated. In other words, the type of box can not be changed when updating.

With picture boxes *T12-*T14 the Tag *$ will be read as the picture path; with text boxes *T3–*T5 as the boxes text; with lines *T0–*T1 it will be skipped.

The type of special boxes *T4 and *T13 is specified in detail with the Tag {48} *Q for the box shape, page: 65.

Polygon picture boxes are exported and imported as *T12 with the polygon description in {70} *t[...].
Polygon text boxes are exported and imported as *T3 with the polygon description in {70} *t[...].

With polygons the box shape is always described by *t[...]; the values *T3 and *T12 only describe the box contents: text or picture.
The box envelope is specified by the pairs of values in *t[...].

Boxes with content “None”

In QuarkXPress apart from the box contents text and picture there is the content type “None”. In these types of boxes neither text nor pictures can be placed in QuarkXPress. DATAform-XTension exports these types of boxes as empty picture boxes. When importing or updating the content these boxes will be placed as empty picture boxes.

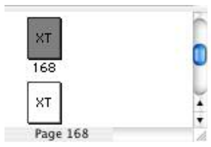
{6} *p Page

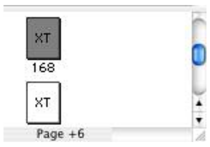
[0......]

This Tag defines the page number upon which the object will be placed or the page that will be created.

Page number in QuarkXPress

The Tag *p describes the page number in the current document; every document, every project or QuarkXPress file, begins with page 1. If the section start is changed, the page number on the left bottom edge of the QuarkXPress window or at the page layout pallette of QuarkXPress varies. This way you can easily read the document page in the current project:





The footer of the page layout pallette shows the chapter page of the selected thumbnail, page 168 in the left picture. Option-click (Alt-click) on the thumbnail shows the actual document/project page, page 6 in the right picture. (A click beside the thumbnails shows the number of pages in the document)

Creating new pages

If the page in the QuarkXPress document, into which the text with DATAformTags should be imported is missing, the page or pages will be newly created. If a master page is specified with the command {3} *P, it will be used for the new pages.

¶*T14*p20*P"Master"*$¶

creates all the missing pages up to page 20 each time with the master page “Master” and on page 20 a round picture box.

Placement on the current page

If in an object the page Tag *p is missing or if it is *p0 (zero), the object will be placed on the current page. The current page is the one which is shown as outlined on the page layout pallette:

*p0 and *px

Objects with page numbers (*px) and page number zero (*p0 or missing Tag *p) can be mixed together in one process. The following lines create one box each on pages 5 and 6 and two boxes on the page that was current when the command was given.

¶*T3*p0*$Current page¶*T3*p6*$Page six¶*T3*p5*$Page five¶*T3*p0*x200*X400*$again current page¶

The Tag *p0 provides the basic function that allows a layouter to work interactively with the database. It allows him to search in the database for the objects needed in the current QuarkXPress page, placing them in QuarkXPress and then continue working with them.

The Tag *px (with page number) is mainly used for the strongly structured catalogues.

{7-10} *xXyY Coordinates

{7}	**x*	*Box position left*
{8}	**X*	*Box position right*
{9}	**y*	*Box position top*
{10}	**Y*	*Box position bottom*

These four Tags define the position and size of an object.

Null point

The null point of the page is independent of the QuarkXPress ruler and page margin settings. It is, when starting the programme, the top left hand corner of the page.

For importing objects the null point can be shifted by the Tags *{61-62} *x>y> => Null point*, see page: 75.
When exporting an object into a text with DATAformTags the object position will always be shown in relation to the pages null points (top left corner).

Accuracy

The information is given in points, with a maximum of three possible decimal places in Quark-XPress. Extra decimal places are rounded up/down upon import. Up to 5 it is rounded down. From 6 on it is rounded up.
¶*T3*x0*X100.5557*$Test¶
creates a box, which is 100.556 points wide. Upon exporting this box the value given in Quark-XPress, 100.556, is used.

Information in the QuarkXPress dialogues

The measurement information given in the QuarkXPress dialogues relates to the ruler null point. The values of the DATAformTags are only shown in the dialogues if the ruler null point is set to the page null point. The ruler null point can be reset to the page null point, by clicking on the ruler´s intersection at top left.
To compare the position details of an imported text with DATAformTags, with the details in the QuarkXPress dialogues, the offset of the import null point must be taken into consideration and added to the position Tags. (See under *{61} *x> Offset > right and {62} *y> Offset > bot-tom*, page: 75)

The width of a box given in the QuarkXPress dialogue is calculated with right minus left value, the height with bottom minus top value:
Box width = {8}*X - {7}*x
Box height = {10}*Y - {9}*y

For the positioning of lines, the following applies:

{7}	*x	left-upper endpoint from the top
{9}	*y	left-upper endpoint from the left
{8}	*X	right-bottom endpoint from the top
{10}	*Y	right-bottom endpoint from the left

Positioning on the pasteboard – objects touch no pages

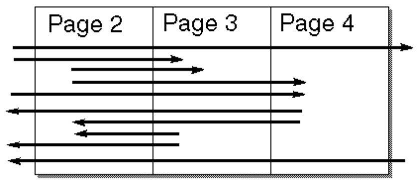
Objects can be placed on the pasteboard, left, top and bottom (but not right). They are placed on the pasteboard of the page specified in *p.
Objects whose coordinates are on the right pasteboard are placed inside the page.

¶*T3*x-300*X-200*p33*$Pasteboard left¶
creates a text box on the left of page 33.

The positioning on the pasteboard is mainly used for exposing additional information about the print job underneath the page.
Objects that lie completely on the pasteboard receive the page number *p0 upon export. The object is therefore not assigned to a page and will be placed on the pasteboard of the current page upon import.

Objects that extend over a page edge.

Lines or boxes that extend over the edge of a page are upon export assigned to the smallest page.

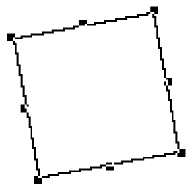


All lines in the above illustration are upon export assigned to the page *p2 and are placed upon import again as shown.

{11} *W Angle

[-360…360]

This Tag defines the angle with which the box/line should be turned in an anti-clockwise direction.
¶*T3*W10,5*$¶
creates a text box that is turned 10.5 degrees to the left.

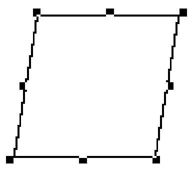


A negative angle will turn the object clockwise. (Tip: pictures may additionally be turned inside their box by *{56} *w Picture angle*, page: 71)

{12} *S Box skew

[-75...75]

This Tag defines the angle with which the box is skewed to the right.
¶*T3*S10*$¶
creates a box that is skewed 10 degrees to the right.



The box skew can be between –75 and 75 degrees.
Tip: The skew of a picture in a box can be set by *{57} *s Picture skew*, page: 71.

{13} *C Columns

[1…30]

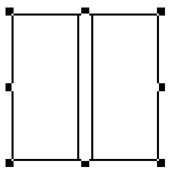
{14} *c Gutter Width

[3…288]

Both these Tags define the columns and the gutter width in a text box.

¶*T3*C2*c10*$¶

creates a text box with two columns 10 points apart.



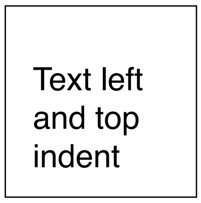
{15-18} *iIjJ Text inset

*{15} *i Text inset left*
*{16} *I Text inset right*
*{17} *j Text inset top*
*{18} *J Text inset bottom*
in each case [3…288]

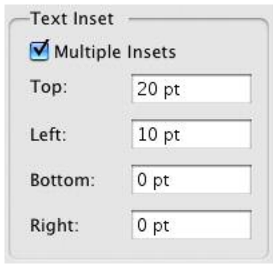
These four Tags define, in points, the text indent in a box.

¶*T3*i10*I0*j20*J0*$Text left and top indent¶

Creates a text box with a left indent of 10 and a top indent of 20 points.



The QuarkXPress check box “Multiple Insets” will be checked by DATAformXTension automatically if the four indents vary.



Text Inset

☑ Multiple Insets

Top:	20 pt
Left:	10 pt
Bottom:	0 pt
Right:	0 pt

{19} *N Supress object printout

[0;1]

This Tag defines, whether the object should be printed or not.

*N1	Supress printout.
*N0	Print object.

With picture boxes the printing of the picture can be suppressed independently from the printing of the box. (see under *{50} *n Supress picture printout*, page: 67.)

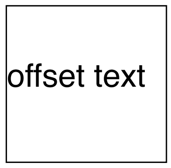
{20} *f First baseline offset

[0…46]

This Tag defines, in points, the offset of the first baseline in a text box.

¶*T3*f30*$offset text¶

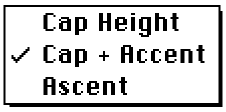
creates a text box with a first baseline offset of 30 points.



{21} *H Cap Height
{22} *h Accent

in both cases [0;1]

Both these Tags control the minimum offset of the first baseline of a text box. The Quark-XPress text box dialogue shows a pull down menu with three options:



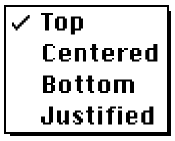
*h0*H1	Cap Height
*h1*H1	Cap and Accent
*h0*H0	Ascent

Both these Tags are only used together and in the above three combinations.

{23} *D Vertical alignment type

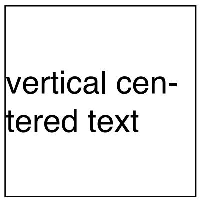
[0..3]

This Tag defines the vertical alignment of the text in a text box. The QuarkXPress text box dialogue shows a pull down menu with four options:



*D0	Top
*D1	Centered
*D2	Bottom
*D3	Justified

¶*T3*D1*$vertical centered text¶
creates a text box with centered text.

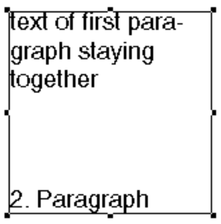


{24} *d Inter ¶ maximum

[0…1080]

This Tag defines the maximum distance between paragraphs of texts, which are vertically justified.
The Tag is only meaningful for text boxes with this property. The DATAformTag for it is *D3.
A large value for *d means that lines inside a paragraph stay together.

¶*T3*D3*d100*$text of first paragraph staying together
2. Paragraph¶


text of first paragraph staying together

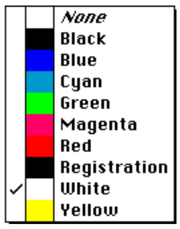
2. Paragraph

{25} *B Box colour

[0…8 then user defined] or ["ColourName"]

This Tag defines the background colour of text and picture boxes.
Colours can, in general, be transferred as colour number or "ColourName"

QuarkXPress originally contains 9 colours that are shown in alphabetical order in the QuarkXPress pull down menu:



Newly created colours are also alphabetically sorted into the pull down menu.

Colour numbers

Colours are numerically assigned to the following *B values:

*B0	White
*B1	Black
*B2	Red
*B3	Green
*B4	Blue
*B5	Cyan
*B6	Magenta
*B7	Yellow
*B8	Registration
*B9	New Colour

Newly created colours can be called by *B9, *B10 etc. The use of colour numbers has the advantage of language independence – the numbers of the base colours are also valid in a German QuarkXPress version, for example.

The following text with DATAformTags creates a palette of the 9 base QuarkXPress colours:

```
¶*T3*y0*Y20*p0*b1*-0*B0*$¶*T3*y20*Y40*p0*b1*-0*B1*$¶*
T3*y40*Y60*p0*b1*-0*B2*$¶*T3*y60*Y80*p0*b1*-0*B3*$¶*
T3*y80*Y100*p0*b1*-0*B4*$¶*T3*y100*Y120*p0*b1*-0*B5*$¶*
T3*y120*Y140*p0*b1*-0*B6*$¶*T3*y140*Y160*p0*b1*-0*B7*$¶*
T3*y160*Y180*p0*b1*-0*B8*$¶
```

Colour names

The box colour can also be specified as a colour name, after the Tag *B. The name must be given in inverted commas (inch signs).

¶*T3*-0*b1*R1*B"Black"*$¶
creates a black text box. The spelling of the name is not case sensitive.

DATAformXTension transfers colours upon export into a DATAformTag file as colour names.

The box colour is shown only if the box is not transparent – the Tag for this is *-0. (See under *{27} *- Transparent*, page: 50)

Translating the nine QuarkXPress base colours

In an English QuarkXPress document/programme colour names are English by default, in a German document, they have German names. DATAformXTension translates, if necessary, the following colour names and assigns them the corresponding colour:

Blau	Blue
Cyan	Cyan
Gelb	Yellow
Grün	Green
Magenta	Magenta
Paßkreuze = Farbmarken	Registration
Rot	Red
Schwarz	Black
Weiß	White

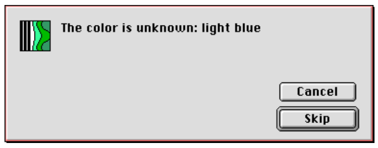
With this function colours are always correctly assigned if their name is given in one of those languages.

¶*T3*-0*b1*R1*B"Registration"*$¶*y110*Y210*T3*-0*b1*R1*B"Paßkreuze"*$¶

creates two boxes in QuarkXPress each with the colour "Registration".

Missing colour messages

Colours are only imported correctly if they have previously been defined in the QuarkXPress document into which they will be placed. If the colour is missing in the document, DATAformXTension shows an error message and the entire import procedure can be aborted.


The color is unknown: light blue

The colour "light blue" is not defined in the QuarkXPress document.
“Cancel” stops the entire import procedure.
“Skip” skips this colour assignment and uses a standard colour.
The standard colours are black for lines, frame styles and text, white for everything else.

If one of the following colours is missing, a message appears: box colour, colours of coloured pictures, colour blends, line and frame colours and the text colour of the box font.
Font colours in the text that are imported by XPressTags are not checked.

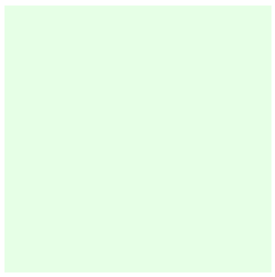
{26} *b Shade

[0…1]

This Tag defines the shade of the box background. The value 1 corresponds to 100%

¶*T3*-0*B"Green"*b0,1*$¶

creates a text box with a 10% green background.



{27} *- Transparent

[0;1]

This Tag defines the transparancy of the box colour. The Tag overwrites as *-1 an additionaly specified background colour. colours are only applied to opaque, meaning non-transparent, boxes. If the Tag *-1 is transferred, the box receives the colour “None” in the QuarkXPress colour menu.

*-0	opaque box with a colour.
*-1	transparant box, i.e. the colour “None”, is checked in the colour menu.

If a box should be coloured, you transfer *-0. If the Tag is missing, the settings in the Quark-XPress tools preferences are used.

Box font

The 10 Tags {28} up until {37} define the box font of a text box. The box font is the font of a new box which is manually drawn in QuarkXPress. If you write in the box, the text will use this font. If various fonts are used in the box, the properties of the first character in the box count as the box font.

{28} *F Font mode

[0;1;2]

This Tag chooses one of three (four in QuarkXPress Windows) possible variants, to treat the font and style properties of new and updated text boxes.
If the Tag is missing, the font will be treated by *F2.

***F0 Update contents retains the box font.**

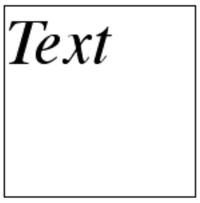
When updating the previous QuarkXPress box font will be used, i.e. a change of the box font in QuarkXPress will be retained when updating. With *F0 all the text placed in the box will take the properties of the first character in the box. The DATAform command “Update contents” overwrites the old text, but uses the previous box font.

***F1 Replace box font**

With this value the box font of text boxes is replaced via the Tags *1 up until *9. New boxes are created with this box font. The command “Update content” updates the box font.

¶*F1*1"Times"*22*324*T3*$Text ¶

creates a text box and sets the text to 24 pt times italic.

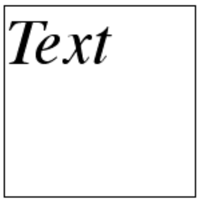


***F2 Apply XPressTags**

With *F2 the XPressTags contained in the text are interpreted.
If a text contains XPressTags, *F2 must be transferred. Otherwise the XPressTags will appear in the text and won´t be translated into font and style properties.
If text boxes are exported including XPressTags (See under DATAformXTension “Prefer-ences...”.), the Tag *F will be transferred as *F2 again otherwise *F1 is exported. (See in appendix under “Text with XPressTags”, page: 132)

¶*T3*F2*$<*f"Times"z24PI>text ¶

delivers the same result as the above example with *F1.



*F3 Interpreting RTF –QuarkXPress for Windows only

With *F3 the RTF instructions included in the text are interpreted.
If a text contains RTF instructions, *F3 must be transferred or the RTF instructions will appear in the placed text and won´t be translated into font and style properties.

RTF format can only be imported. An export of text is possible as plain text or text with XPressTags, but not in RTF format.

The RTF text must have been created by a programme that is supported by QuarkXPress for Windows (for more details see the QuarkXPress manual) i.e. Word for Windows; an RTF text that was created in Word for Macintosh may eventually be unreadable in QuarkXPress for Windows.

Box fonts or XpressTags?

The use of box fonts is only possible if the entire box text is to be set in the same font. The box font is an advantage if the font style is changed in QuarkXPress and the text only and not the font style should be overwritten when updating. In all other cases you will work with XPressTags and style sheets.

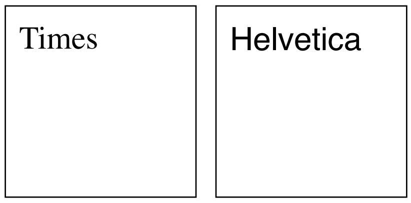
{29} *1 Font

["Name"] or [Font number]

This Tag defines the box font of a text box. Parameters can either be a "FontName", in inverted commas, or a font number. Upon exporting the box the font name will always be transferred.

*121 is equivalent to *1"Helvetica"
*120 is equivalent to *1"Times"
It is suggested that you use the font name for transfers.

¶*T3*F1*1"Times"*$Times¶*T3*x110*X210*F1*1"Helvetica"*$Helvetica¶
creates two text boxes each with a different font.



If the font name is not found, the style sheet “normal” is used when working with XpressTags and a system font is used when working with the box font.

¶*T3*i5*F2*$<*f"xxTimesxx">Text¶*T3*x110*X210*i5*F1*1"xxTimesxx"*$Text¶
creates two boxes with incorrect font calls.

The text with DATAformTags for the first box contains an incorrect font name in the XpressTag (*f"xxTimesxx") and for the second an incorrect box font (*1"xxTimesxx").

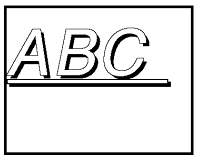
{30} *2 Font style

[0; 20…6; 29…12]

This Tag defines the font style of the box font. The style is transferred as a number, whereby various styles may be added together.
Example: 1 (bold) + 2 (italic) makes 3 = bold-italic, i.e. the Tag ***2**3
The standard styles in QuarkXPress have the following numbers.

0	standard
1	bold
2	italic
4	underline
8	outline
16	shadow
32	superscript
64	subscript
512	strike through
1024	all caps
2048	small caps
4096	word underline

¶*T3*F1***2**94***3**24*$ABC ¶
creates a text box with the font style 94.



Style number 94 is the sum of the styles 2+4+8+16+64.

{31} *3 Font size

[2…720]

This Tag defines the size of the box font in points.
***3**7.25 fixes the box font size to 7.25 pt.

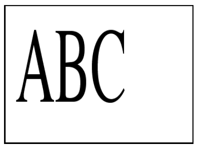
{32} *4 Font horizontal scale

[0,25…4]

This Tag defines the width of the box font. The non-scaled value is 1, i.e. 100%. A value between 0,25% and 400% or ***4**0.25 and ***4**4 is possible.

¶*T3***3**40.5*F1*1"Times"***4**0.5*h1*H1*$ ABC¶

creates a text box with a 50% wide “ABC” in 40.5 pt.



The same box can be created with XpressTags and *F2, by:
¶*T3*F2*$<*f"Times"z40.5Ph50> ABC¶

{33} *5 Font/picture colour

[0…8 then user defined] or ["ColourName"]

colours can in general be specified by colour number or by "ColourName".
The standard QuarkXPress colours range from *50 (or *5”white”) to *58 (or *5"registration").
The definition of colours is explained in detail in *{25} *B Box colour*, page 48.

Text boxes
This Tag defines the font colour of the box font in a text box. If a coloured text is placed in QuarkXPress by DATAform, a colour with that name must exist in the current document.

Picture boxes
This Tag defines the colour of greyscale or b/w pictures. QuarkXPress can colour certain picture formats, see the relevant QuarkXPress reference manual. Pictures receive a colour from the menu style/colour. If a coloured picture is placed in QuarkXPress by DATAform, the colour with this name must already exist in the document.
If Tag *5 is missing, the picture will not be coloured.

¶*T12*5"Blue"*$Omega.tif¶	Creates a picture which is coloured blue.
¶*T12*$Omega.tif¶	Places the picture with normal greyscale values.
¶*T12*5"xBluex"*$Omega.tif¶	DATformXTension 4 reports the missing colour "xBluex".

{34} *6 Font/picture shade

[0…1]

Text boxes
This Tag defines the shade of the box font in text boxes. 1 means 100%

¶*T3*350*F1*1"Times"*51*60,3*h1*H1*$ S¶

creates a text box with a 30% grey “S” in 50 pt.

Picture boxes
This Tag defines the shade of the picture colour in picture boxes. QuarkXPress can colour greyscale and b/w pictures and additionaly can modify the shade in b/w pictures, see the QuarkXPress reference book.
If the Tag *6 is missing, the shade of the picture will not be adjusted and will be 100%.

{35} *7 Picture negative

[0; 1]

This Tag defines in picture boxes whether the picture should be displayed as positive or negative. QuarkXPress can display TIFF pictures and certain PICT pictures as negative, see QuarkXPress reference manual.

*71	the picture is displayed as negative
*70	or missing Tags; the picture will be displayed normally postive.

¶*Z3*T12*5"Blue"*60.7***71***$Omega.tif¶

shows a 70% shaded blue coloured picture in negative:

{35} *7 Font kern amount

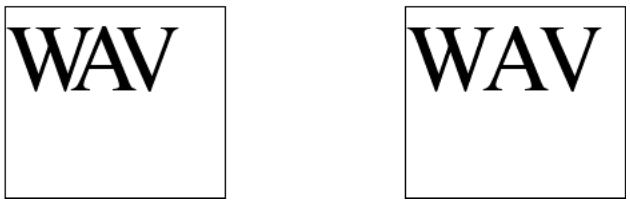
{36} *8 Font track amount

in both cases [-500…500]

Both Tags define in text boxes (for picture boxes, see above) the character spacing in the box font. For both Tags the following applies: positive values increase the spacing, negative values decrease it. The difference is only: font kerning defines the spacing between characters, font tracking defines the distance of a character itself to the right. A value for the font kerning is only applied, if a box contains text.

¶*T3*335*F1*1"Times"*7-25*h1*H1*$WAV¶

creates a text box, the text “WAV” appears in Times 35 point and is kerned by 25 units. One unit is 1/200 of an Em space, the absolute value varies with the font. The right box shows the same text without font kerning.



{37} *9 Font baseline shift

[-3...3]

This Tag defines the font baseline shift of the box font.
The Tag moves the baseline in multiples of the font size. Positive values move the characters down, negative up. In the QuarkXPress dialogues the shift is shown with inverted signs. The value changes with the font size.
With a size of 12 pt ***9**1.5 shifts downwardly by 12 x 1.5 = 18 pt.

¶*T3*F1*312***9**1.5*$Text with baseline shift¶

creates a text box with a baseline shift of 18 points down.

Text with
baseline shift

In the QuarkXPress typography dialogue the offset appears with this font size as –18 pt.

Frames and lines

The four Tags {38} up until {41} define the properties of text and picture box frames and lines: type, width, colour, shade. The arrowheads of lines are defined by the Tag *{58} *E Arrowheads*, cf. page: 72.

{38} *L Frame/line style

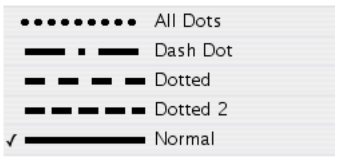
Frames [1…100; 124…134], lines [0…4; 24…100; 128…133]

With text and picture frames this Tag defines the frame style in relation to the frame style dialogue and with lines the line style in relation to the line style dialogue.

Frame styles

By default QuarkXPress has some predefined frame styles; further frame styles can be created in the frame style editor. The predefined frame styles are called with *L124 up until *L134, the predefined bitmap boxes for square shapes as well as your own box styles with *L1 up until *L100.

Example: QuarkXPress offers these five simple frame styles *L124 - *L128:



The following text creates five boxes with the shown frame styles:

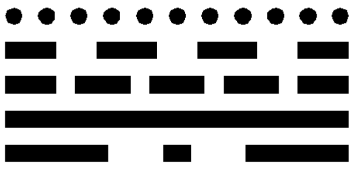
```
¶*T3*L127*l5*y0*Y90*$¶*T3*L126*l5*y100*Y190*$¶*T3*L124*l5*y200*Y290*$¶*
T3*L125*l5*y300*Y390*$¶*T3*L128*l5*y400*Y490*$¶
```

Line types

The five frame styles in the above example are available as line styles by *L0 – *L4. Predefined are further the lines *L128 – *L133. Self created lines range from *L24 to *L100.
The five above frame styles are created as lines by:

```
¶*T1*L4*l5*x10*X10*$¶*T1*L1*l5*x20*X20*$¶*T1*L2*l5*x30*X30*$¶*
T1*L0*l5*x40*X40*$¶*T1*L3*l5*x50*X50*$¶
```

The text creates the following block out of 5 lines:



Line	Frame
*L4	*L127
*L1	*L124
*L2	*L125
*L0	*L128
*L3	*L126

The right column shows the *L values for the same lines as frame styles.

The easiest way to find out the *L value of a shape is by exporting the related frames/line as text with DATAformTags.

{39} *l Frame/line width

[0…864]

This Tag defines the width of the frame in text or picture boxes as well as the width of lines in points.

{40} *M Frame/line colour

[0…8 then own] oder ["ColourName"]; with two colours ["ColourName";"ColourName"]

This Tag defines the colour of box frames as well as the colour of lines. Colours can be transferred as a colour number or as "ColourName". The standard QuarkXPress colours range from *M0 (or *M"white") to *M8 (or *M"registration".) The definition of colours is explained in detail at Tag *{25} *B Box colour*, page: 48.

¶*T0*L0*E1*l6*M2*$¶

Creates a red arrow

Lines and frames with gaps can have a second gap colour. In this case both colours are transferred as names in square brackets.

¶*T0*L128*E0*l8*M["black";"cyan"]*$¶

Creates a black line with a blue gap

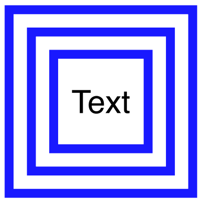
{41} *m Frame/line shade(s)

[0…1]; with two values [shade; shade]

This Tag defines the shade of box frames and lines. 100% is equivalent to 1.

¶*T3*F2*L134*l20*M"Blue"*m0.9*D1*$<*C>Text¶

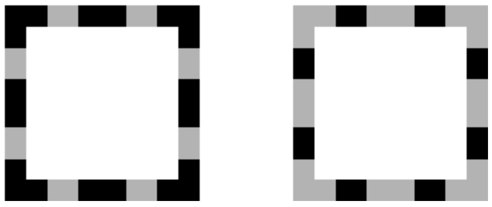
creates a text box with the frame style No. 134, 20 pt wide, blue, a shade of 90% and vertical (by DATAformTag *D1) and horizontal (by XPressTag *C) centered text.



Some frame and line types can have a special gap colour. The shade values for these colours are transferred between square brackets, like in *m[1;0.3]:

¶*T3*L124*l8*M["black";"black"]*m[1;0.3]*$¶*
T3*L124*l8*M["black";"black"]*m[0.3;1]*$¶

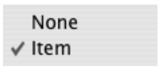
Creates to boxes with dashed lines 100% / 30% black and 30% / 100% black.



{42} *R Runaround type

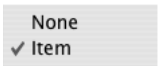
Text boxes [0-3]; lines [1;2]; picture boxes [0–8]

This Tag defines the way in which a text should Runaround an object.
With text boxes *R defines the runaround popup menu as well as the check box “Run Text Around All Sides”. It results in four combinations:



*R1	None	+	...All Sides is OFF
*R0	Item	+	...All Sides is OFF
*R3	None	+	...All Sides is ON
*R2	Item	+	...All Sides is ON

For lines these options of the runaround popup menu are supported:



*R1	None
*R0	Item

With picture boxes these options are available:

*R1	None
*R0	Item
*R2	Auto Image
*R4	Embedded Path
*R5	Alpha Channel
*R6	Non-White Areas
*R8	Same As Clipping
*R3	Picture Bounds

There are not all options available with all types of pictures. Some settings require a rescan of the picture by QuarkXPress. The “Rescan” is not carried out by DATAformXTension.

{43-47} *ruUvV Runaround

*{43} *r Runaround all*
*{44} *u Runaround left*
*{45} *U Runaround right*
*{46} *v Runaround top*
*{47} *V Runaround bottom*
in each case [-288…288]

These five Tags define the distance in points with which a text should Runaround the object.

With picture and text boxes four separate values or the total value *r are possible with *R0 (Runaround boxes). The total value is used on import only if the separate values are missing or set to zero. If one of the single values *u, *U, *v or *V is other than zero, the single values are used in all four cases.

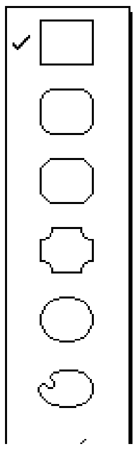
In pictures boxes with *R2 (auto outline) as well as lines with *R0 only the total value *r is possible.

{48} *Q Box shape

*only with *T4 and *T13; [0;1;2]*

This Tag defines the corner shape of box types *T4 and *T13. In the QuarkXPress sub-menu “Item/Shape” 6 box shapes may be chosen.
Rectangles and ovals are described by the box types *T3/*T5 and *T12/*T14. (see also under *{5} *T box type*, page: 37.)

For the corner shape of box types *T4 and *T13 three variants can be chosen: convex, straight and concave.

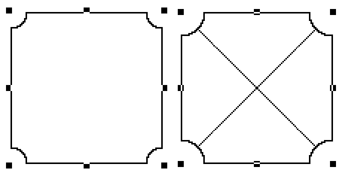


Text box		**Picture box**	
*T3	right text box	*T12	right picture box
*T4*Q0	convex corners	*T13*Q0	convex corners
*T4*Q1	straight corners	*T13*Q1	straight corners
*T4*Q2	concave corners	*T13*Q2	concave corners
*T5	oval text box	*T14	oval picture box
*T3*t[...]	polygonal text box	*T12*t[...]	polygonal picture box

The illustration shows the QuarkXPress sub-menu Item/Shape shape with the relevant DATAformTags.

¶*T4*Q2*q20*$¶*T13*Q2*q30*$¶

creates one text and one picture box with concave corners



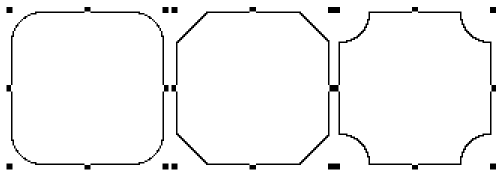
{49} *q Corner radius

*only with *T4 and *T13; [0…288]*

This Tag defines the corner radius or the slope for the box shapes *T4 and *T13. The value *q is only used with both these box shapes. The diameter is given in points. In the QuarkXPress dialogue the radius is shown as half of the value *q.

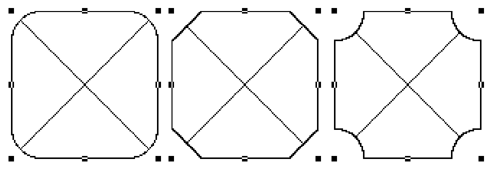
¶*q40*T4*Q0*$¶*q40*T4*Q1*$¶*q40*T4*Q2*$¶

creates the three possible *T4 text boxes *Q0, *Q1, *Q2, with a corner radius of 40 pt.



¶*q40*T13*Q0*$¶*q40*T13*Q1*$¶*q40*T13*Q2*$¶

creates the three possible *T13 picture boxes *Q0, *Q1, *Q2, with a corner radius of 40 pt.



{50} *n Suppress picture printout

[0;1]

This Tag defines whether the picture should also be printed when printing out. The pictures boxes and their backgrounds are not affected by this command (See also under *{19} *N Suppress object printout*.)

*n0	Print picture
*n1	Suppress picture printout

{51} *Z Picture position =>

[1..4]

The Tag *Z allows the automatic scaling and positioning of a picture within its box when importing. (On updating the *Z-Tags are not applied, the properties of the picture and the box remain unchanged.)

*Z1 Centre picture in box. The Tags *{54-55} *()Picture offset* are overwritten by the calculated values. This function can be performed from QuarkXPress with the contents tool and command+shift+M.

*Z2 Fit picture to box. The Tags *{52-53} *Kk Scale picture* and *{54-55} *() Picture offset* are not applied. This function can be performed from QuarkXPress with the contents tool and command+shift+F.

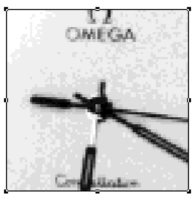
Z3 Fit, but scale proportionally. The Tags for scaling (*K, *k) and the offset *(and *) are not applied. This function can be performed from QuarkXPress with the contents tool and command+option+shift+F.

*Z4 Fit box to picture. The Tags for scaling (*K, *k) are applied, the tags for the offset *(and *) are not applied.

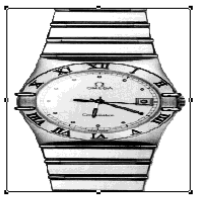
¶*T12*Z1*$:MyPicture¶*T12*Z2*$:MyPicture¶*
T12*Z3*$:MyPicture¶*T12*Z4*K0,3*k0,3*$:MyPicture¶

creates four picture boxes with the same picture and the four possible values for *Z:

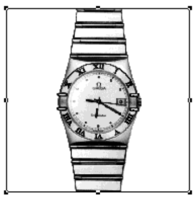
*Z1 centred



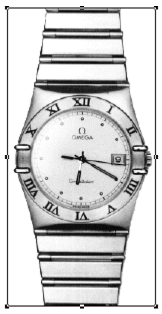
*Z2 adjusted



*Z3 prop. adjusted



*Z4 box adjusted



{52-53} *Kk Scale picture

*{52} *K Picture scale across*
*{53} *k Picture scale down*
in each case [0,1…10]

The Tag *K defines the picture scale across. *K1 is equal to a scale of 100%. Possible values are *K0.1 up until *K10, i.e. from 10% up until 1000%.

The Tag *k defines the picture scale down. *k1 is equal to a scale of 100%. Possible values are *k0.1 up until *k10, i.e. from 10% up until 1000%.

¶*T12*K0,40*k0,40*$:MyPicture¶

creates a picture box, loads the picture “MyPicture” from the QuarkXPress documents folder and scales the picture evenly to 40%:

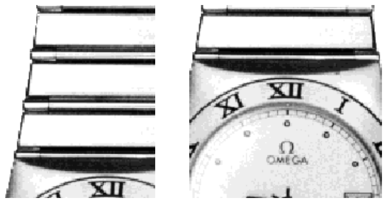
{54-55} *() Picture offset

*{54} *(Picture offset across*
*{55} *) Picture offset down*
in each case [relative to picture and box size]

Both Tags define the horizontal and vertical offset of the picture within the picture box, in points. Positive values shift the picture to the right/down.

¶*T12*K0,55*k0,55*$:MyPicture¶*(-18*)-37*T12*K0,55*k0,55*$:MyPicture¶

loads a picture twice into new boxes, the first without offset, the second with a horizontal offset of –18pt and a vertical offset of 37 pt.



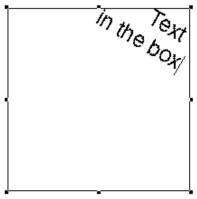
{56} *w Picture/text angle

[-360…360]

This Tag defines the rotation of a picture/text within the box in an anti-clockwise direction. Negative angles turn the box content clockwise. Tip: The rotation of the box itself is specified by the Tag *{11} *W Box angle*.

¶*w20*Z1*T12*K0,55*k0,55*$:MyPicture¶*T3*w-30*$Text in the box¶

loads "MyPicture" from the document folder scales it to 55% and rotates it 20˚ anti-clockwise. Rotates the text in the second box 30˚ clockwise



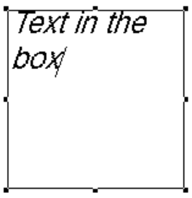
{57} *s Picture/text skew

[-75…75]

This Tag defines the angle with which a picture or a text in a box is skewed to the right. Negative angles skew the picture to the left. Tip: The skew of the box itself can be set with *{12} *S Box skew*.

¶*s-20*Z1*T12*K0,55*k0,55*$:MyPicture¶*T3*s20*$Text in the box¶

loads “MyPicture” from the document folder scales it to 55% and skews the picture 20˚ to the left. Skews the text in the second box 20˚ to the right.

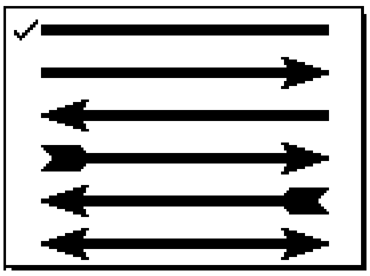


{58} *E Arrowheads

[0…5]

This Tag defines the type of arrowheads. The following 6 types are predefined in QuarkXPress and are called by the Tag *E:

*E0
*E1
*E2
*E3
*E4
*E5



¶*l5*T0*E5*$¶

creates a 5 pt thick double arrow.
The remaining line properties are defined by the Tags {38}-{41}.

{59} *§ Locked

[0;1]

This Tag defines whether the created object should be locked. A locked object can not be moved by the mouse or have its size changed; it can be unlocked by the QuarkXPress command “Item/Unlock”.

*§1	Lock object.
*§0	Don’t lock object.

¶*T3*§1*$¶*T12*§1*$¶*T0*§1*$¶

creates a text box, a picture box and a line, all as locked objects.

{60} *Δ (*Delta) Adjust box height =>

[0…6]

This Tag allows the automatic enlarging/shrinking of text boxes in relation to the quantity of text when placing and updating in QuarkXPress.
The maximum box enlargement up or downwards will be limited by the page margins.

The Tag *Δ has the following functions:

*Δ0	or missing Tag, the box size remains unchanged.
*Δ[1-3]	move lower box edge, the Tag {10} *Y is overwritten.
*Δ1	fit lower box edge to the text.
*Δ2	move the lower box edge upwards if the box is too big.
*Δ3	if a Runaround occurs, move the lower box edge downwards.
*Δ[4-6]	move upper box edge, the Tag {9} *y is overwritten.
*Δ4	adjust upper box edge to the text.
*Δ5	move the upper box edge downwards if the box is too big.
*Δ6	if a Runaround occurs, move the upper box edge upwards.

The function *Δ1 (adjust lower box edge to text) is manually called in QuarkXPress with command+*.

This Tag is only used in the direction text with DATAformTag => QuarkXPress, i.e. not exported from QuarkXPress.

¶*T3*#7*Δ1*$one line¶

creates a standard text box and shrinks it to fit the text.

one line

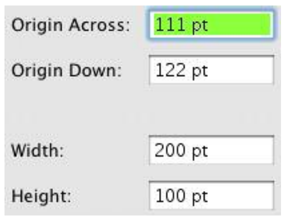
{61-62} *x>y> Null point =>

*{61} *x> Null point offset across =>*
*{62} *y> Null point offset down =>*

Both these Tags set the import null point of the DATAformXTension to the given value. When starting the programme, the import null point lies on the page null point, which is on the top left corner of the page. Positive values adjust the null point to the right or downwards. The values are shown in points. The offset is added to the values of the Tags *x, *y, *X and *Y upon import. (See also under position Tags *{7-10} *xXyY coordinates*, page: 39)

¶*T3*x100*y100*X300*Y200*x>11*y>22*$shift null point¶*T3*x100*y100*X300*Y200*$applied null point¶

creates two text boxes with the same coordinates:



Origin Across:	111 pt
Origin Down:	122 pt
Width:	200 pt
Height:	100 pt

The first object shifts the null point. The offset applies to the object itself and all subsequent placements.
The second object contains no offset Tags. If the offset Tags are missing in an object, its coordinates refer to the import null point.

The import null point is valid until it is replaced with a new offset Tag or by a new programme start.
New offset Tags always define the new import null point in relation to the page null point – they are not added to an old import null point.
The values *x>0 and *y>0 reset the import null point to the page null point.

In each case the last setting of the import null point remains the same for all import procedures and document editing until QuarkXPress is quit.

The import null point applies only for the import of objects. The Export of objects into a text with DATAformTags always refers to the page null point.

{63} *+ Section page <=

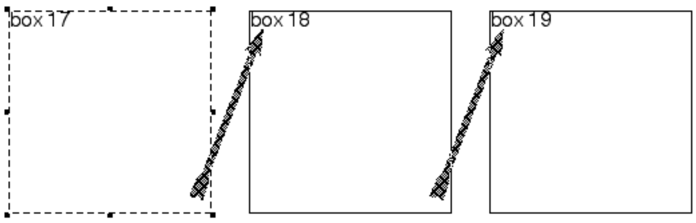
When exported from QuarkXPress this Tag contains the QuarkXPress section page on which the box is placed.
This Tag is only used in the direction QuarkXPress => database.
The section page is the page which is shown in the lower left part of the QuarkXPress window. The section page can be defined in QuarkXPress by the “section start” function, independently from the document page.
A section page can be preceded by up to four letters in QuarkXPress, e.g. “P. 193”, the page number is preceded by “P. ”.
In QuarkXPress the section page is automatically shown in a text box by the function “page number” (command-3). The exported section page consists of the number including the preceding characters as the example shows:
*+"P. 193".
The DATAform 4D interface deletes the inch signs and transfers the value P. 193 into the array line DX_Array{63}.

{64-65} *»« Chain

*{64} *» Chain: next box*
*{65} *« Chain: previous box*

Both these Tags make the chaining of text boxes possible.
» is the ASCII character 200 (MacOS: option-shift-Q, Windows: Alt+0200);
« is the ASCII character 199 (MacOS: option-Q, Windows: Alt+0199)
The DATAform object No. of the next box comes after *»; after *« comes the object No. of the previous object. *»0 means: this box has no successor, it is the last in the chain. *«0 means: this box is the first in the chain.

¶*«0*#17*»18*T3*x0*X100*F2*$box 17<\b>box 18<\b>box 19¶*«17*#18*»19*T3*x120*X220*$b¶*«18*#19*»0*T3*x240*X340*$c¶



The text with DATAformTags creates three chained text boxes. The text is transferred in full with the first box. By the two expressions <\b>, the XPressTag for "next box", the text is distributed amongst all three chained boxes.

«0#17*»18	Box #17 is the first in the chain (*«0), it is chained to #18 (*»18)
«17#18*»19	Box #18 is in the middle, its predecessor is #17 (*«17), its successor is Box #19 (*»19)
«18#19*»0	Box #19 is the last in the chain (*»0), its predecessor is #18 (*«18).

The chain's complete text is always transferred with the first box of the chain and is exported the same way. If a chain's middle box is exported, the result is a box information, but no text. If the contents of chained boxes are updated, only the text of the first box is updated.
If chained text boxes are updated, the complete text of the chain is replaced by the new text. The text is therefore transferred with the first box in the chain. Text in further DATAform objects in the chain is skipped when updating and importing.

Building up chains

Boxes that will be chained must be transferred via an import procedure.
Boxes that are already in documents are ignored when the chain is built at the end of the import procedure; they are not included in the chain, and you receive an error message upon import.

The same example in the 4D interface:
DX_Array{64} contains the box ID of the next box in the chain.
DX_Array{65} contains the box ID of the previous box in the chain.
The last box in the chain has the value DX_Array{64} = "0"
The first box in the chain has the value DX_Array{65} = "0"

1. boxes in the chain:
```
DX_Array{1}:="box 17<\c>box 18<\c>box 19"
DX_Array{2}:="17" `boxID = 17
DX_Array{5}:="3"  `text box
DX_Array{64}:="18" `chained with the box No. 18
DX_Array{65}:="0" `no preceding box
```

2. boxes in the chain:
```
DX_Array{1}:=""
```

```
DX_Array{2}:="18" `boxID = 18
DX_Array{5}:="3"  `text box
DX_Array{64}:="19" `chained with the box No. 19
DX_Array{65}:="17" `previous boxes No. 17
```

3. boxes in the chain:

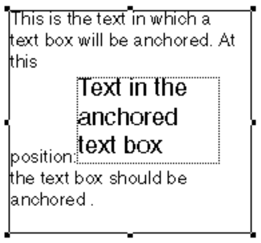
```
DX_Array{1}:=""
DX_Array{2}:="19" `boxID = 19
DX_Array{5}:="3"  `text box
DX_Array{64}:="0" `chain with no further boxes
DX_Array{65}:="18" `previous box No. 18
```

{66-67} *Aa Anchor

*{66} *A Anchor in box*
[object number]

This Tag defines the destination box in which the text box or the picture should be anchored. The object number is a DATAform object number. The destination box must be positioned before the anchored box in the text with DATAformTags.
The box will be anchored in the destination boxes text. The place holder is in the text of the destination box at the anchoring point {{object number of the anchored box}}. The place holder consists of {{, the object number of the box that should be anchored at this position and closes with }}.

¶*T3*x0*y500*X120*Y650*39*F1*#20*$This is the text in which a text box will be anchored. At this position:{{33}} the text box should be anchored .
¶*T3*X70*Y42*A20*#33*$Text in the anchored text box¶


This is the text in which a text box will be anchored. At this
Text in the anchored text box
position: the text box should be anchored .

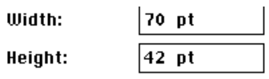
The text with DATAformTags creates this text box with an anchored box.

Procedure

At the end of the import procedure, DATAformXTension meets the box number #33 and the Tag *A20. It then looks for a text box with object number #20, the destination box for anchoring, and then looks in the text of the box for the place holder “{{33}}” and anchors the box at this position.
When exporting anchored boxes the reverse occurs and instead of the anchored box the place holder is inserted in the text. The anchored boxes are then exported following the “mother box”.

Box size

Only two position details are required for anchored boxes:



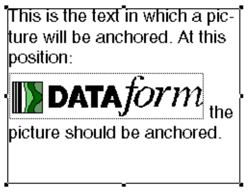
Width:	70 pt
Height:	42 pt

The width is transferred by the DATAformTag {8} *X
The height is transferred by the DATAformTag {10} *Y
Both the other position Tags {7} *x and {9} *y may be left out or must be set to zero.

Example

The following text with DATAformTags creates a picture box of 100 x 23 pt and anchors it in a text box:

¶*T3*x0*y500*X120*Y590*39*F1*#1517*$This is the text in which a picture will be anchored. At this position: {{1518}} the picture should be anchored.¶*T12*X100*Y23*A1517*#1518*Z3*$DATAformLogo¶


This is the text in which a picture will be anchored. At this position: the picture should be anchored.

The picture is adjusted proportionally to the box. Before trying place the picture “DATAform-Logo” in your QuarkXPress folder.

Anchoring in chained boxes

If the destination box is chained to other boxes, the first box of the chain must precede the anchored boxes. With chained boxes the destination box is always the first box of the chain. The entire text of the chain is also transferred with this box.

Tip: If you create very long chains with many anchored boxes, switch on Off-screen drawing in the QuarkXPress preferences.

Update anchored boxes

Anchored boxes may be updated individually as if they were independent boxes.
Create both following boxes:

¶*T3*x0*y500*X120*Y650*39*F1*#1520*$ This is the text in which a text box will be anchored. The text box should be anchored at this position:
{{1521}}.¶*T3*X70*Y42*A1520*#1521*$the text in the anchored text box¶

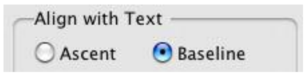
Then copy the following text and choose the command “update contents”:

¶*T3*X70*Y42*#1521*$new text¶

The anchored box receives a new text.

*{67} *a Anchor: align with text*
[0;1]

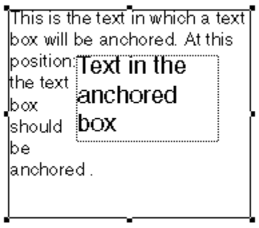
This Tag defines the text alignment of an anchored box. There are two possibilities:


Align with Text
Ascent [✓ Baseline]

*a0 or missing Tag	align to baseline
*a1	align to ascent

The following text with DATAformTags creates the same box as in the previous examples, but the anchored box is aligned to ascent:

¶*T3*x0*y500*X120*Y650*39*F1*#1522*$This is the text in which a text box will be anchored. At this position:{{1523}} the text box should be anchored .¶*a1*T3*X70*Y42 *A1522*#1523*$Text in the anchored box¶


This is the text in which a text box will be anchored. At this position: Text in the anchored box the text box should be anchored .

{68} *| Flipping

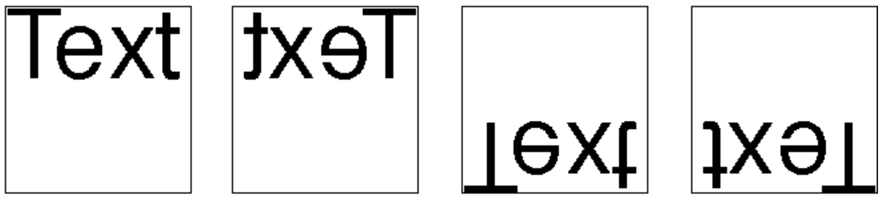
[0;1;2;3]

This Tag defines the flipping of text and pictures. The symbol “|” is created on the Macintosh by option-7; it has the ASCII value 124.
The following values are possible:

*|0 or missing Tag	don’t flip
*|1	flip horizontally
*|2	flip vertically
*|3	flip horizontally and vertically

¶*|0*T3*F1*336*$Text¶*|1*T3*F1*336*$Text¶*
|2*T3*F1*336*$Text¶*|3*T3*F1*336*$Text¶

creates four text boxes with the possible flips:



*|0 don’t flip *|1 horizontal *|2 vertical *|3 horizontal and vertical

Underneath each text box the corresponding illustration of the QuarkXPress measurement palette is shown.

{69} *e Blending; *modified in DATAformXTension 7.2.1*

[…]

This Tag defines a blend. Between the square brackets there are 9 values separated by semi-colon, i.e.:
*e[1364738882;27672;0;2;"green";0;0.9;45;0]
From DATAformXTension 7.2.1 a 10th parameter saves the opacity.

Only a few of these values are fixed definitions, the remaining values should be transferred back to QuarkXPress as they were received.

1.	1364738882	the XTension ID, in this case from CoolBlends
2.	27672	blend ID, in this case Mid-linear blend
5.	"green"	colour name of the second colour of the blend
7.	0.9	shade 90%
8.	45	angle 45 degrees
10.	0.64	opacity 64%

The first colour of the blend is the normal background colour; it is defined by *B and the shade by *b.

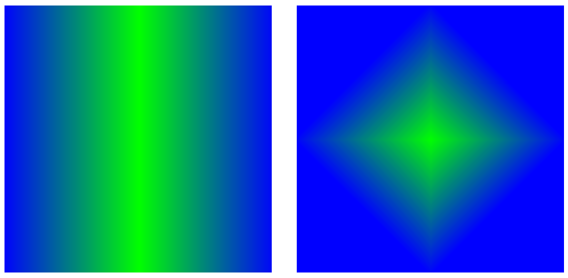
¶*T3*-0*B"cyan"*b1*e[0;10000;3;2;"yellow";0;1;0;0]*$¶*
T3*-0*B"cyan"*b0.6*|0*x200*X300*e[0;10000;3;2;"yellow";0;0.9;45;0;0.47]*$¶

creates a text box with a colour blend of 100% cyan (*B"cyan"*b1) to 100% yellow, not twist-ed and a text box with a colour blend of 60% cyan (*B"cyan"*b0,6) to 90% yellow turned by 45 degrees with 47% opacity.

*-0 checks that the box is not transparent otherwise the blend is not displayed. In this example the XTension ID is zero; the internal QuarkXPress linear blend, without CoolBlends, will be used. (You may see the colours in the PDF version of this manual.)

¶*T3*-0*B"blue"*b1*e[1364738882;27672;0;2;"green";0;1;0;0]*$¶*
T3*-0*B"blue"*b1*|0*e[1364738882;27674;4;2;"green";0;1;0;0]*$¶

creates two picture boxes with colour blends with the CoolBlends XTension. XTension must be loaded.



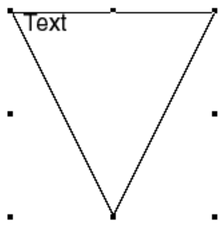
{70} *t Polygon

[…]

This Tag defines a polygonal text or picture box. The X and Y values of the individual polygon points, separated by a semi-colon, appear between the square brackets. Bezier curves from are not supported.
The smallest polygon defines a triangle; it consists of a start point, two further points and an end point. The end point must be specified separately and must be the same as the start point – polygons are always closed. The smallest polygon therefore contains four points that are defined by eight values. The details are given in pt.
A text polygon is defined by *T3, a picture polygon by *T12.
Other values for *T are not allowed with polygons.

```
¶*T3*x100*y100*X200*Y200*t[100;100;200;100;150;200;100;100]*$Text¶
```

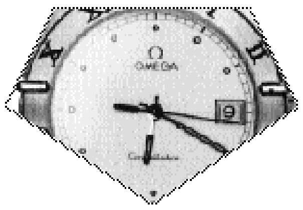
creates a triangle as a text box. With *x100*y100*X200*Y200 the surrounding rectangle of the polygon is defined. The polygon must fit into the surrounding rectangle exactly.

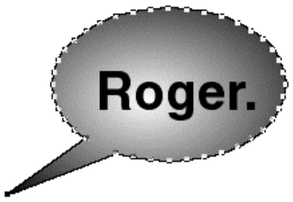


The eight anchor points of a polygon lie exactly on the surrounding rectangle.

```
¶*T12*x75*y100*X225*Y200*t[100;100;200;100;225;150;150;200;75;150;100;100]*
Z1*K0.8*k0.8*$:MyPicture¶
```

creates a pentagon, loads the picture “MyPicture” from the QuarkXPress folder, scales it to 80% and places it in the centre of the box, left picture.





```
DfXT+2.0¶*#0*G0*T3*p0*x19.92*y406.005*X158*Y498.746*W0*S0*C1*c12.024*i0*I0*
j0*J0*f0*H0*h0*D0*d0*B"black"*b1*F1*1"Helvetica"*21*326*41*5"black"*61*70*80*
90*L128*l0*M"black"*m1*R1*r0*Q0*q271.694*»0*«0*§0*N0*
-0*+"29"*|0*e[1364738882;27676;4;2;"white";0;1;0;0]*
t[159;445;158.286;450.944;156.162;456.742;152.679;462.25;147.925;467.334;142.015;
471.868;135.095;475.74;127.336;478.856;118.929;481.138;110.081;482.531;101.01;
482.999;91.937;482.533;83.088;481.142;74.68;478.861;20.92;499.746;59.999;471.875;
54.087;467.342;49.331;462.26;45.847;456.752;43.72;450.955;43.004;445.012;43.716;
439.068;45.837;433.271;49.317;427.762;54.069;422.678;59.976;418.143;66.893;414.269;
74.651;411.153;83.056;408.869;91.904;407.475;100.975;407.005;110.047;407.471;118.895;
408.86;127.302;411.139;135.062;414.251;141.984;418.121;147.897;422.652;152.654;
427.734;156.141;433.24;158.269;439.036;159;445]*$ Roger.¶
```

creates a polygon with a Circular Blend from black to white, rechtes Bild.

{71} *@ Character set table

[0; 1; 2]

Allows the creation of a platform independent text with DATAformTags. DATAformXTension converts all character set dependent texts from the Macintosh character set into the Windows ANSI character set and vice versa.

The Tag *@ is defined for each DATAform object and specifies the character set in which the text as well as the colour names etc. are written. The XTension converts, if necessary, the character set into that of the current platform. DATAformXTension also converts all the names that are contained in XPressTags, such as colour names or style sheet names. Style sheet names e.g. may, unlike a simple text export/import with XPressTags, keep any umlauts if used on various platforms.

If DATAformXTension exports a box, it will always export the Tag *@. It shows the database from which platform the text with DATAformTags comes from. It is then up to the database to properly convert the text.

In principle the "transmitter" of a text with DATAformTags sends in the Tag *@, denoting the character set encoding its text. The "receiver" converts the text if necessary.

*@0	or missing Tag: character set is not specified; no conversion
*@1	character set encoding is MacOS
*@2	character set encoding is Windows-ANSI

DATAformXTension converts the text depending on the current platform:

If the XTension is running under QuarkXPress Windows and receives an object with *@1, the text will be converted from Mac into Windows ANSI. If it receives an object with *@0 or *@2, no conversion is performed.
If the XTension is running under QuarkXPress Mac and receives an object with*@2, the text will be converted from ANSI into Mac. If it receives an object with*@0 or*@1, no conversion is performed.

DATAform 4D interface

The character set conversion is taken over entirely by the DATAform 4D interface. Under Mac OS, Windows and equally in platform independent use, the character set is converted automatically.

(The interface always exports the Tag *@1 as 4D always works internally with the Mac character set. When importing text with DATAformTags all text, colour names etc. are converted if necessary.)
(See also the procedure DX_Information in the *DATAform 4D interfac*, page: 121)

{72} *± Trapping

[…]; “own value”/”user defined” [-36 pt…36 pt]

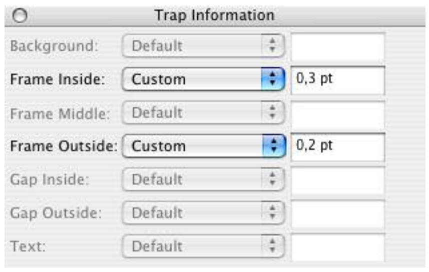
This Tag allows the import and export of the trapping properties of pictures, text boxes and lines. The properties that are exported and imported are:

- “Frame inside” and “Frame outside” in text and picture boxes with frames,
- “Background” in text and picture boxes without frames,
- “Line” in lines.

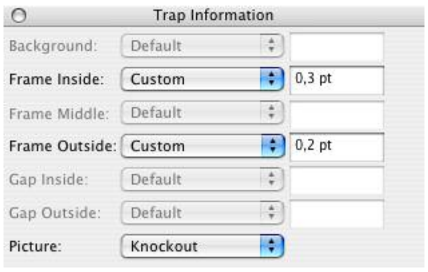
Limitations
- The trapping properties of text i.e. single characters are not supported. The XPressTags filters in version do not support text trapping properties.
- The trapping properties of the second colour of lines and frame styles are not supported.

Trapping properties examples:

Boxes with frames


Trap Information

Background:	Default	
Frame Inside:	Custom	0,3 pt
Frame Middle:	Default	
Frame Outside:	Custom	0,2 pt
Gap Inside:	Default	
Gap Outside:	Default	
Text:	Default	

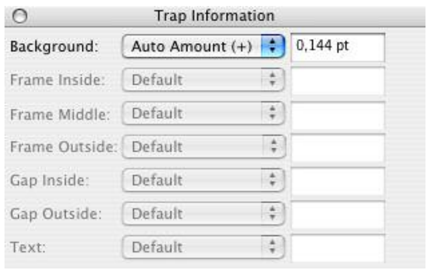

Trap Information

Background:	Default	
Frame Inside:	Custom	0,3 pt
Frame Middle:	Default	
Frame Outside:	Custom	0,2 pt
Gap Inside:	Default	
Gap Outside:	Default	
Picture:	Knockout	

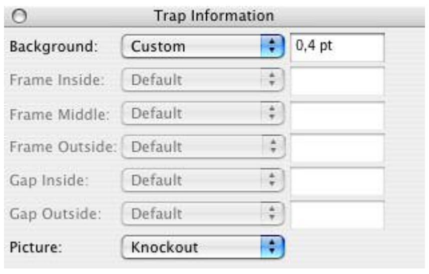
The illustration shows the trapping palette for boxes with frames. These settings lead with the text box shown left hand to the Tag *±[1;0;5;5;0;0.3;0.2]; with the picture box right hand to the Tag *±[1;0;5;5;0;0.3;0.2;2]
With text boxes, the square brackets always contain seven parameters, with picture boxes always eigth, activated depending on the box type:

1. version ID, at this time, always 1.
2. pull down menu line “Background”, numbered from 0 to 5 (not activ for this box).
3. pull down menu line “Frame Inside”, numbered from 0 to 5, in this example 5.
4. pull down menu line “Frame outside”, numbered from 0 to 5, in this example 5.
5. custom value for “Background”, in this case 0 (not activ for this box).
6. custom value for “Frame Inside”, in this example 0.3 pt.
7. custom value for “Frame outside”, in this example 0.2 pt.
8. pull down menu line “Picture”, numbered from 0 to 2 (activ with the right box only).

Boxes with background (without frames)


Trap Information

Background:	Auto Amount (+)	0,144 pt
Frame Inside:	Default	
Frame Middle:	Default	
Frame Outside:	Default	
Gap Inside:	Default	
Gap Outside:	Default	
Text:	Default	

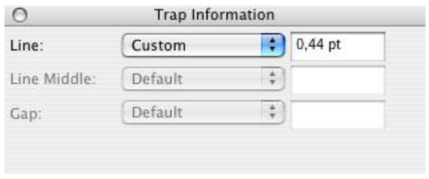

Trap Information

Background:	Custom	0,4 pt
Frame Inside:	Default	
Frame Middle:	Default	
Frame Outside:	Default	
Gap Inside:	Default	
Gap Outside:	Default	
Picture:	Knockout	

The left illustration shows the trapping palette for a box with background, but without a frame; the DATAformTag is: *±[1;3;0;0;0;0;0]; the fourth line “Auto Amount (+)” is activated in the pull down menu; the second paramter in the Tag therefore shows 3.
The right illustration shows a picture without frame; the picture is set to knockout; the DATA-

formTag is: *±[1;5;0;0;0;0;0;2]; the eighth parameter shows 2.

Lines


Trap Information
Line: Custom 0,44 pt
Line Middle: Default
Gap: Default

With lines the square brackets contain three parameters only:
*±[1;5;0.44]
1. version ID, at this time always 1.
2. pull down menu line, numbered from 0 to 5 for “Line”, in this case 5.
3. custom value for „Line“, in this case 0.44 pt

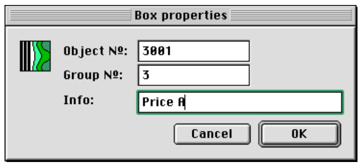
{73} *? Infotext

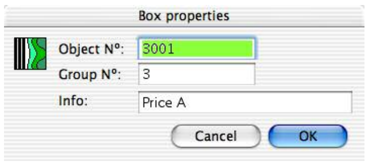
["Text" max. 30 characters]

The tag allows you to save a string of characters with a box or line. When the box is exported into a DATAformTags-text, the string will be exported after *?. The developer may use the tag to store information not fitting in the tags *{4} *G* or *{2} *#* . The enduser may use it e.g. to name boxes.

¶*#3001*G3*T12*?"Price A"*$¶

creates a picture box with the Infotext “Price A”. The Infotext can be edited in the dialogue “DATAform/Box properties...”:

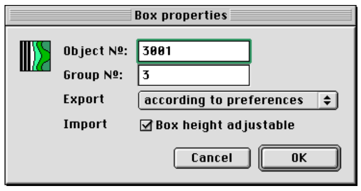

Box properties
Object Nº: 3001
Group Nº: 3
Info: Price A

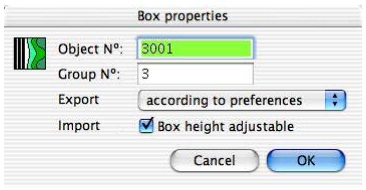

Box properties
Object N°: 3001
Group N°: 3
Info: Price A

By the tag *{75} *g Box properties* (see page: 89) you can set the field Infotext to “read only” or “invisible” too, for example:

¶*#3001*G3*T3*?"invisible Info"*g"222202"*$the text in the box¶

Creates a text box with an invisible Infotext. The text “invisible Info” is not shown in the DATAform Box properties dialogue:


Box properties
Object Nº: 3001
Group Nº: 3
Export: according to preferences
Import: ☑ Box height adjustable


Box properties
Object N°: 3001
Group N°: 3
Export: according to preferences
Import: ☑ Box height adjustable

When the box is exported as a DATAformTags-text, an invisible Infotext will always be exported unchanged: ¶*#3001*G3*T3*p4*@1 ... *?"invisible Info"*$Text in the box¶
An invisible Infotext is not converted into the Mac or ANSII character set, neither on import nor on export. The Tag *{71} *@* is not applied.

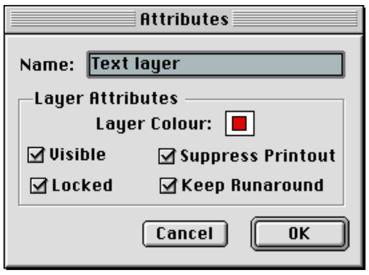
{74} *= Layer

[…]

The Tag informs about the layer in which the box is located. On import the box is moved again to the layer with the same name. Missing layers will be created on import.

The Tag contains 5 parameters:
*=["LayerName";Visible;Locked;Suppress Printout;Keep Runaround]

They reflect the options of the Layer dialogue in QuarkXPress 5:



Attributes

Name: Text layer

Layer Attributes

Layer Colour: 🟥

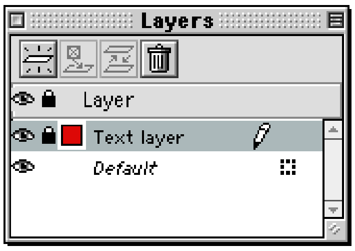
☑ Visible ☑ Suppress Printout

☑ Locked ☑ Keep Runaround

Name:	max. 32 characters
Visible:	1 = layer is visible; 0 = invisible. The value is ignored on import.
Locked:	1 = locked layer; 0 = the items can be moved
Suppress Printout:	1 = do not print; 0 = print out
Keep Runaround:	1 = also invisible items cause runaround; 0 = ignore invisibles

¶*T3*p1*=["Text layer";1;1;1;1]*$Hello¶

Creates a text box with the text “Hello” on page 1 and positions the box in the layer “Text layer”. If a layer “Text layer” does not exist in the document, it will be created.



If a layer is invisible during import, it will be changed into “visible”.

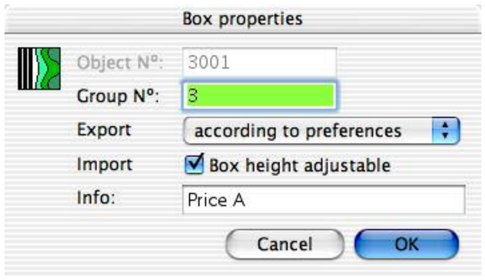
The “Default” layer is neither exported nor imported.

If the Tag *= is missing in an object description, it is placed in the last current layer.

{75} *g Box properties

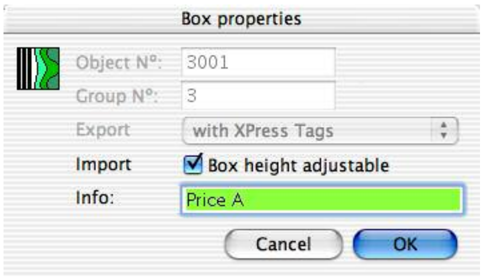
["6 digits"]

The Tag allows importing and exporting the DATAform properties of a QuarkXPress item. The Tag exports and expects 6 digits always. The default value on missing *g Tag is *g"222222". The single digits define the possibilities in the dialogue „DATAform/Box properties...“, as well as the function while duplicating boxes. The Tag *g gives enhanced controls to the database developer and more options for individualizing box properties to the enduser.


Box properties

Object N°: 3001
Group N°: 3
Export: according to preferences
Import: ✓ Box height adjustable
Info: Price A

Cancel | OK


Box properties

Object N°: 3001
Group N°: 3
Export: with XPress Tags
Import: ✓ Box height adjustable
Info: Price A

Cancel | OK

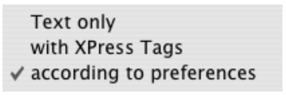
1. digit “Object N°.” [Values: 2 or 5]
    2 = The field “Object N°.” can be edited.
    5 = The field “Object N°.” is in read only mode.

    ¶*#3001*G3*T3*?"Price A"*g"522222"*$¶

    creates a text box with properties as shown top left. Its “Object N°.” cannot be edited.

2. digit “Group N°.” [Values: 2 or 5]
    2 = The field “Group N°.” can be edited.
    5 = The field “Group N°.” is in read only mode.

3. digit “Export with XPressTags” [Values: 0, 1, 2 as well as 5, 6, 7]
    The digits enables a box specific export with XPressTags. It sets the popup menu:


Text only
with XPress Tags
✓ according to preferences

    0 = “Text only”, the text should be exported without XPressTags, as plain ASCII text.
    1 = The text of the box should be exported including XPressTags.
    2 = The current definition in the dialogue “DATAform/preferences” should be used. The checkbox in this dialogue ☐ Include XPress Tags is effective for all boxes set to “according to preferences”. This is the default setting for manually created boxes or if the Tag *g is missing.
    The values 5–7 set the popup menu analogous to 0–2 and disable it additionally; plus 5 means deactivate the item.

    ¶*#3001*G3*T3*?"Price A"*g"556222"*$¶

    creates a text box which will be exported with XPressTags always – regardless wether this is set in DATAform/preferences. And the setting “with XPressTags” is not editable; the 3rd digit is 1 + 5 = 6. See picture top right.

    With picture boxes and lines the digits 3 and 4 will be ignored, but still exported and imported.

4. digit “Box height adjustable” [Values: 0 , 2 as well as 5, 7]
   The digit enables the enduser to directly disable box adjustment, box by box, for the next import of the box. The database can save the setting and reuse it for the next placement.

   0 = box height will not be adjusted to the amount of text on next import; the value of *{60} *Δ (*Delta) Adjust box height*, page: 74, will be ignored.
   2 = box height will be modified according to *{60} *Δ Adjust box height*.

   The values 5–7 set the check box analogous to 0–2 and disable it additionally; plus 5 means deactivate the item.

   ¶*#3001*G3*T3*?"Price A"*g"556722"*$¶

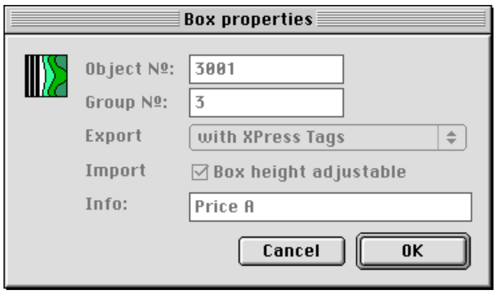
   creates a text box, which will be adjusted on import according to *Δ, set by the database. The check box is ON, but cannot be modified, since the 4th digit is 2 + 5 = 7.

5. digit “Infotext” [Values: 0 , 2, 5]
   The digit defines the properties of the Infotext field. (The content of this field is exported and imported by *{73} *? Infotext*.)

   0 = the field is invisible; Infotext will be exported unchanged.
   See the example at *{73} *? Infotext*, page: 87.
   2 = the field can be edited.
   5 = the field is in read only mode.

   ¶*#3001*G3*T3*?"Price A"*g"556752"*$¶

   creates a text box with a read only Infotext; furthermore all other fields of the dialogue are in read only mode:

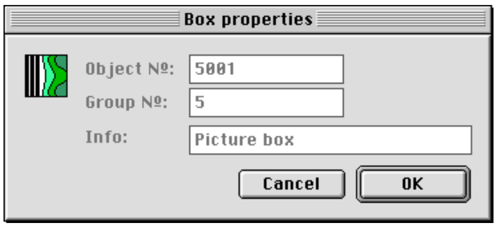

Box properties
Object Nº: 3001
Group Nº: 3
Export: with XPress Tags
Import: ☑ Box height adjustable
Info: Price A

6. digit “Duplicating” [Values: 0 , 1, 2, 3]
   The digit controls the treatment of the DATAform object Nº. while duplicating a DATA-form box. (There is no corresponding field in the properties dialogue.)

   0 = Set the object Nº to Zero while duplicating the box.
   1 = Set the object Nº to Zero and put it as Infotext if it is an empty string.
   2 = Duplicate the box as it is, whithout changes.
   3 = Set the object Nº to Zero and put it as Infotext always, overwriting an old Infotext.

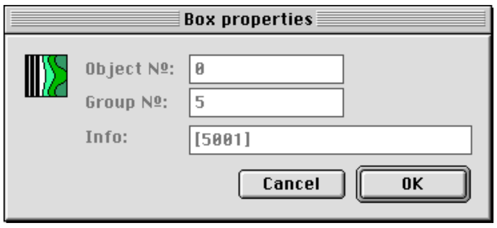
   ¶*#5001*G5*T12*?"Picture box"*g"552253"*$¶

   creates a picture box with these box properties:


Box properties
Object Nº: 5001
Group Nº: 5
Info: Picture box

   The 6th digit in Tag *g reads 3, i.e. “Set the object Nº to Zero and put it as Infotext always”

If the enduser duplicates this picture box, the new picture box will get these box properties:



Object Nº:	0
Group Nº:	5
Info:	[5001]

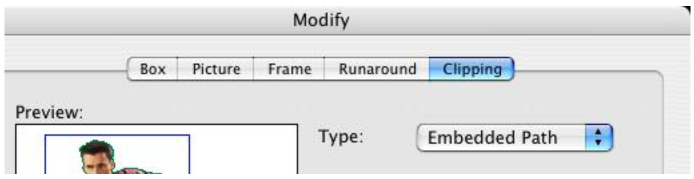
- The object Nº is set to Zero (it was 5001 original).
- The original object Nº is saved as Infotext [5001].
  [5001] can be read as: “A copy of box 5001”.
  The original Infotext is overwritten.
- The group Nº and all other box properties are copied unchanged.

These functions of resetting the object Nº are called in all cases of duplicating one or more boxes, such as
- copy and paste,
- duplicate, step and repeat,
- drag and drop to another QuarkXPress document,
- drag thumbnail pages to another document.

{76} *z Picture clipping, from DATAformXTension 6

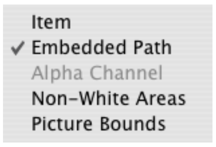
Picture box [0; 3–6]

The Tag defines the type of picture clipping as adjustable in the Modify dialogue, Tab “Clipping” in the popup menu “Type”:



Type: Embedded Path

The popup menus “Type” shows up to five options according to the picture type:

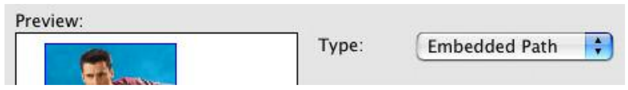


*z0	Item
*z4	Embedded Path
*z5	Alpha Channel
*z6	Non-White Areas
*z3	Picture Bounds

(The numbering is analogous to *{42} *R Runaround type*, page: 63.)
There are not all options available with all types of pictures. Some settings require a rescan of the picture by QuarkXPress. The “Rescan” is not carried out by DATAformXTension.

¶*T12*z4*$:HKO.eps¶

Loads the picture with the setting “Embedded Path”:



Type: Embedded Path

By a click on “Rescan” the path is applied and you will get the illustration shown above.

{77} *, Additional Page =>, from DfXTension 6.5; DfPlugin 4.0.0 (11)

["1";"0"]

The tag takes care that an additional page exists after the target page. Thereby facing pages can be filled from the left page.

*,"1"	creates an additional page after target page if missing.
*,"0"	or missing tag, does nothing.

¶*T3*p2*,"1"*$Box on page 2¶*T3*p2*,"1"*x800*X900*$Box on page 3¶

In a new document with facing pages a new page will be created and the first box will be placed on page 2, the target page. Due to *,"1"an dditionla 3rd page will be created. Caused by its big x values the second box will be placed on this page 3. The second *,"1" won't create another page since the additional page for page 2 already exists.

The tag works in the direction DATAform tags text => layout program only. It is not exported by DATAformXTension.

{78} */ Drop Shadow, from DATAformXTension 7.2.1

[...]

The Tag allows importing and exporting an items drop shadow. Picture, text boxes and lines may have a drop shadow.
14 values reside between the square brackets, e.g.:

*/[14;33,552;111;5,93;0,77;1,14;45;0,76;1;1;0;0;0;"black"]

Drop Shadow
Drop Shadow

1.	14	Count of parameters, in this version 14 always.
2.	33,552	Distance in pt
3.	111	Angle [-180° bis 180°]
4.	5,93	Blur [0 pt bis 144 pt]
5.	0,77	Opacity [0 bis 1], 1 entspricht 100%
6.	1,14	Scale [0,1 bis 10], 1 entspricht 100%
7.	45	Skew [-75° bis 75°]
8.	0,76	Shade [0 bis 1], 1 entspricht 100%
9.	1	Inherit Item's Opacity, ON = 1
10.	1	Synchronize Angle, ON = 1
11.	0	Runaround Drop Shadow, ON = 1
12.	0	Item Knocks Out Drop Shadow, ON = 1
13.	0	Multiply Drop Shadow, ON = 1
14.	"Black"	Drop Shadow's Colour

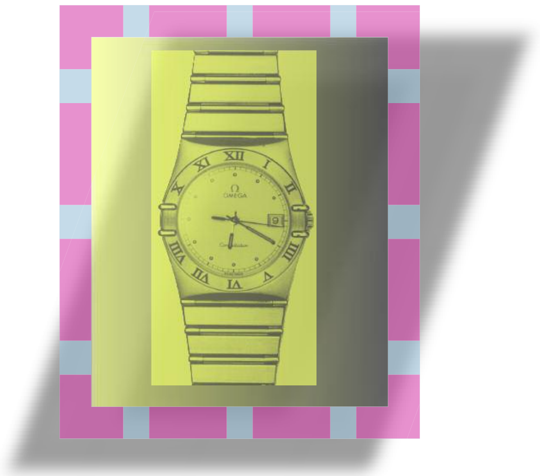
{79} *~ Opacity, from DATAformXTension 7.2.1

[...]

The Tag allows importing and exporting the opacity of an items various colours. Colours with low opacity have a high degree of transparency and vice versa.

6 values reside between the square brackets, e.g.:

*~[6;60;61;62;63;64]



1.	6	Count of parameters, in this version 6 always.
2.	60	Opacity of the box, [0 bis 1], 1 entspricht 100%
3.	61	Opacity of frame or line
4.	62	Opacity of the gap colour of frame or line
5.	63	Opacity of a picture
6.	64	Opacity of a picture background

Further settings of colour opacity exist with blendings and drop shadows. The opacity of these properties is saved with the according tags directly.

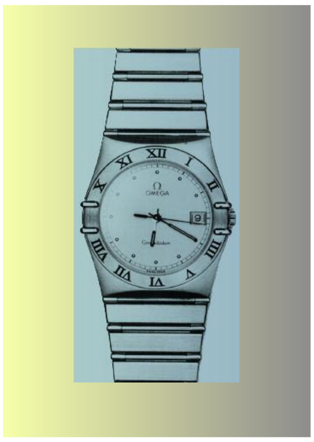
- {69} *e Blending, opacity is supported as a new 10th parameter.
- {78} */ Drop Shadow, the 5th argument defines the drop shadow's opacity.

{80} *0 (*zero) Picture Background, ab DATAformXTension 7.2.1

["Farbe";Tonwert]

The tag defines colour and shade of the picture backgound or gray scale pictures and line arts. The tag marker is *0, asterisk zero.

*0["Cyan";0,8]



Picture background is Cyan with 80% shade.
The opacity of the picture background colour is defined by tag {79} *~ Opacity, see above.

The picture colour is defined by {33} *5 Font/picture colour.

The box colour is defined by {25} *B Box colour and as the case may be the colour of blendings by {69} *e Blending.

III. Functional objects

Functional objects are DATAform objects with a type *T that is greater than 100. They offer functions that go further than normal DATAform objects:

- They set the mode with which the following DATAform objects should be processed such as “import” or “delete”.
- They activate or deactivate the group function.
- They export information about the current QuarkXPress document or create a new document.
- They save the QuarkXPress document or change the page view.

Functional objects may be included in a text with DATAformTags, however, this is not mandatory.
All functional objects work in one direction only. They are not bi-directional like most other DATAformTags. With one exception, all functional objects work in the direction towards QuarkXPress and are therefore not exported.
Only the object **T107 document parameter <=* is exported. It provides information concerning the current QuarkXPress document and is exported as the first object in a text with DATAformTags.
When importing a text with DATAformTags it is skipped and may be omitted without problem.

Box objects and functional objects

Box object	DATAform object which describes a QuarkXPress box or a QuarkXPress line, ¶*T3*$hello¶
QuarkXPress Object	A box or a line in QuarkXPress.
Functional object	DATAform object that, for example, defines the importing mode ¶*T101*M1*$¶

There are two basic modes of processing a text with DATAformTags: import and update. Normally the mode will be defined previous to reading by selecting the corresponding menu commands “Import boxes” and “Update contents” or by the equivalent commands of the message interface “IMPORT” and “UPDATE”.
The chosen mode is in this case used for the entire import procedure.

Functional objects

Functional objects allow you, for instance, to set the processing mode by the text with DATAformTags itself. The mode remains valid until a new functional object is transferred.

Functional objects do not themselves create or describe a QuarkXPress box. They are always positioned before following box objects and define e.g. how these objects should be translated into a QuarkXPress object.

Syntax

Functional objects are made up of the type of function *T with a value >100 and one or more arguments, given under *M etc.

Example: ¶*T101*M1*$¶

The argument Tags, like M, each have their own meaning within functional objects.

The same example as in the DATAform 4D interface:

```
DX_Write (0)   `initialisation

DX_Array{5}:="101"        `*T101 Import
DX_Array{40}:="1"         `*M1 Mode dialogue
$0:=DX_Write (1)          `close object and pass it to the interface

        `Here follow the box objects that should be imported,
        `each one is closed with DX_Write (1)

DX_Write (2)              `end of export
```

Functional objects have priority

Functional objects overwrite the menu functions. If a text with DATAformTags containing the functional object *T101 is imported with the menu command “Update contents”, the boxes will still be imported; the functional object *T101 has priority over the calling method.

Or vice versa: If the text with DATAformTags contains the functional object *T102 for updating, the following text causes an update procedure even if it is called by the menu command “Import boxes”.

In a text with DATAformTags a functional object is valid only until the next functional object. You may switch at will between IMPORT, DELETE and UPDATE.

Demonstration

- Place the picture “DATAform.EPS” into your QuarkXPress folder.
- Set the DATAform preferences in QuarkXPress to import by dialogue.
- Import text with DATAformTags “DATAformShow.QXP”.

Boxes will be imported, changed, deleted, newly created etc. The demonstration shows the possibilities of the functional objects IMPORT, DELETE and UPDATE like in a movie, but uses DATAformXTension only.

*T101 IMPORT =>

¶*T101*Mx*$¶

Sets the processing mode for the following box objects to “IMPORT”

Suppress dialogues

*M defines the treatment of duplicates, i.e. boxes with a DATAform ID, that are already present in a QuarkXPress document.
If needed, the dialogue Cancel/Skip/Replace appears.

*M0	or missing *M Tag, the default settings are valid, see below
*M1	show duplicate dialogue
*M2	skip duplicates
*M3	replace duplicates
*M4	create all objects without checking for duplicates.

Default settings
Calling “Import boxes” from the menu command automatically transfers internally, before loading, the functional object ¶*T101*M1*$¶, i.e., these preferences are each time pre-set.
Calling the function “IMPORT” by message automatically transfers internally before loading the functional object ¶*T101*M2*$¶
If a text with DATAformTags contains no import functional object with *M, it is automatically processed according to the calling method.

Calling in the DATAform 4D interface

```
DX_Write (0)            `initialisation

DX_Array{5}:="101"      `*T101 import
DX_Array{40}:="1"       `*M1 mode dialogue
$0:=DX_Write (1)        `close object and transfer to the interface

        ` Here follow the box objects that should be imported,
        ` each one closed with DX_Write (1)

DX_Write (2)            `end of export
```

*T102 UPDATE =>

¶*T102*Mx*$¶

Sets the processing mode to “update contents” for the following box objects.

Suppress dialogues

M defines the treatment of objects that are not found, i.e. boxes with a DATAform ID, which are not found in a QuarkXPress document.
If needed, the dialogue Cancel/Skip/Replace appears.

*M0	or missing *M Tag, the default settings are used, see below
*M1	show dialogue
*M2	skip objects which could not be found
*M3	create objects which could not be found

Default settings
Calling “update contents” from the menu command automatically transfers internally, before loading, the functional object ¶*T102*M1*$¶, i.e. the preferences are each time pre-set.
Calling the function “UPDATE” by message automatically transfers internally before loading the functional object ¶*T102*M2*$¶
If a text with DATAformTags contains no update functional object with *M, it is automatically processed according to the calling method.

Calling in the DATAform 4D interface

```
DX_Write (0)            `initialisation

DX_Array{5}:="102"      `*T102 update
DX_Array{40}:="1"       `*M1 mode dialogue
$0:=DX_Write (1)        `close object and transfer to the interface

      `Here follow the box objects that should be updated,
      `each one closed with DX_Write (1)

DX_Write (2)            `end of export
```

*T103 DELETE =>

¶*T103*Mx*$¶

Sets the processing mode for the following box objects to “delete”.
Box objects with *#x or *Gx (the x stands for a number other than zero) that are found in the QuarkXPress document are deleted, box objects with *#0 and *G0 (or missing Tags) are skipped.

Suppress dialogue

*M defines the treatment of objects that are not found. (A *Gx object is considered not found, if not a single box with this group number is found.)
If required, one of the following dialogues will be shown: “The box with object No. xxx could not be found” or “No boxes with the group No.xxx could be found.”
The dialogue offers the two buttons Cancel and Skip. Cancel stops the entire process. Skip skips the current box object.

*M0	or missing *M Tag, the default settings are used, see below
*M1	show dialogue
*M2	skip object

Default settings
Calling a load procedure automatically transfers internally before loading the functional object ¶*T103*M1*$¶. This means that this preferences are preset each time.
Calling the load procedure by message automatically transfers internally before loading the functional object ¶*T103*M2*$¶
If a text with DATAformTags contains no import function with *M, it will be automatically processed according to the calling method.

Box objects after ¶*T103*$¶

The box objects that come after ¶*T103*$¶ describe the boxes which should be deleted in the QuarkXPress document. These box objects can contain complete object descriptions, but must have at least three Tags: *T, *$, *# or *G.

Complete example

¶*T101*$¶*#111*T3*$text in box¶*T103*$¶*T3*#111*$¶

places the box #111 in the left/top page corner and immediately deletes it. The text contains the objects:

¶ switch to import mode¶ the box/boxes to be imported¶ switch to delete mode¶ the box/boxes to be deleted¶

The following rules apply for the boxes that will be deleted:

¶*T3*#15001*$¶	Deletes the box with object No.: 15001. The QuarkXPress object must not be a *T3 box; a line or a picture can also be deleted with this method.
¶*T12*G15*$¶	Deletes all boxes in the document with the group No. 15.
¶*T13*G0*#0*$¶	Nothing happens; neither *# nor *G are defined.
¶*T3*$New text...¶	Nothing happens; neither *# nor *G are defined.
¶*T3*G15*#15001*$¶	Only deletes box #15001; not the group G15.
¶*T3*G15*#0*$¶	Deletes all boxes in the document which have the group No. 15.

Calling the function in the DATAform 4D interface

Example 1
Delete all boxes with a specific group number:

```
DX_Write (0)              `initialisation

DX_Array{5}:="103"        `*T103 = DELETE functional object
DX_Array{40}:="1"         `*M1 mode dialogue
$0:=DX_Write (1)          `activate DELETE for all following objects

DX_Array{5}:="3"          `any valid box type, e.g. *T3
DX_Array{4}:="23"         `delete group No. 23, *G23
$0:=DX_Write (1)

DX_Write (2)              `end of export
```

Example 2
Erase two boxes with specific object numbers:

```
DX_Write (0)              `initialisation

DX_Array{5}:="103"        `*T103 = DELETE functional object
DX_Array{40}:="1"         `*M1 mode dialogue
$0:=DX_Write (1)          `activate DELETE for all following objects

DX_Array{5}:="3"          `any valid box type, e.g. *T3
DX_Array{2}:=23001        `delete the box with ID 23001, *#23001
$0:=DX_Write (1)

DX_Array{5}:="3"          `any valid box type, e.g. *T3
DX_Array{2}:=23101        `delete the box with ID 23101, *#23101
$0:=DX_Write (1)

DX_Write (2)              `end of export
```

*T104 DELETE ALL =>

¶*T104*$¶

Deletes all boxes and lines in the current QuarkXPress document.
All objects on all pages, DATAform boxes as well as manually placed objects, will be deleted.

Boxes on master pages are not deleted.
QuarkXPress document pages are not deleted.
The procedure cannot be undone.

Calling in the DATAform 4D interface

```
DX_Write (0)            `initialisation

DX_Array{5}:="104"      `*T104 = delete all boxes
$0:=DX_Write (1)

      `Here may follow the objects which should be placed in the empty
      `QuarkXPress document each one is closed with DX_Write (1)

DX_Write (2)            `end of export
```

*T105 FIT IN WINDOW =>

¶*T105*$¶

Sets the page view of the current QuarkXPress document at the end of the import procedure to "Fit in window".
The functional object has the same effect as the QuarkXPress menu command "Fit in window" in the view menu, command 0.

Calling in the DATAform 4D interface

```
DX_Write (0)                `initialisation

  `box objects may follow here

DX_Array{5}:="105"          `*T105 = fit in window after import
$0:=DX_Write (1)

  `box objects may also follow here

DX_Write (2)                `end of export
```

*T106 GROUPING =>

¶*T106*Mx*$¶

Sets the option “group” in the DATAform settings. The option is only set for the current procedure. The settings in the dialogue itself are not changed.

Example:

¶*T106*M1*$¶	Switch grouping on.
¶*T106*M2*$¶	Switch grouping off.
¶*T106*M0*$¶	Nothing happens.
¶*T106*$¶	Nothing happens.

If a text with DATAformTags contains the object ¶*T106*M1*$¶, the imported boxes will be grouped at the end of the import procedure, even if the user has switched off grouping in the preferences dialogue.

¶*y0*Y100*T3*G1*#1*$boxes 1¶*y110*Y210*T3*G1*#2*$boxes
2¶*y220*Y320*T3*G1*#3*$boxes 3¶*T106*M1*$¶

creates three text boxes of the group *G1 and groups them.

Calling in the DATAform 4D interface

```
DX_Write (0)              `initialisation

      `Here follow the box objects that should be imported or updated

DX_Array{5}:="106"        `*T106 = group functional object
DX_Array{40}:="1"         `*M1 turn grouping on
$0:=DX_Write (1)          `activate group

DX_Write (2)              `end of export
```

The functional object can be placed anywhere within the text with DATAformTags

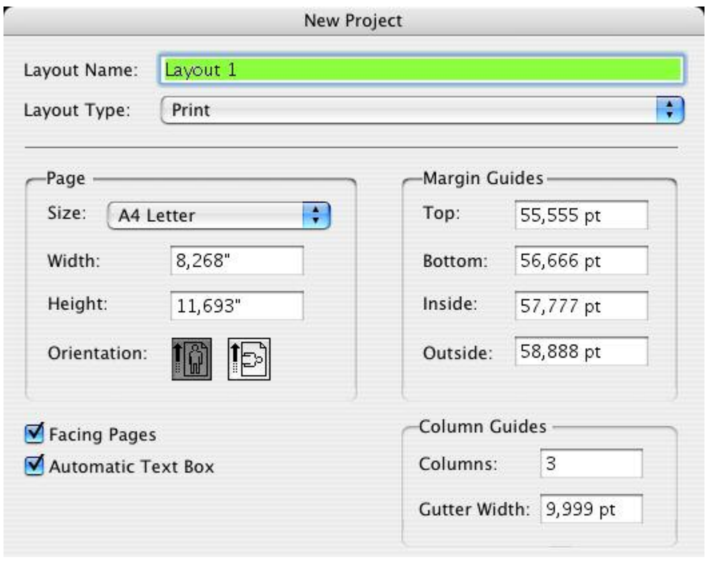
*T107 DOCUMENT PARAMETER <=

¶*T107*e[…]*$¶

Every export command exports a *T107 object providing information about the QuarkXPress project (formerly document) from which it has been exported. The object *T107 is always the first object exported. Your database tests *T for 107 or the DX_Array{5} for "107" and skips the object or receives the document information.

Demonstration

- Choose points as the measuring units in the QuarkXPress preferences with all windows closed; all measurements are exported by DATAform as points and can be compared easily with this setting.
- Create this new document:


New Project

Layout Name: Layout 1
Layout Type: Print

Page
Size: A4 Letter
Width: 8,268"
Height: 11,693"
Orientation:

Margin Guides
Top: 55,555 pt
Bottom: 56,666 pt
Inside: 57,777 pt
Outside: 58,888 pt

☑ Facing Pages
☑ Automatic Text Box

Column Guides
Columns: 3
Gutter Width: 9,999 pt

- Write “hello” in the created automatic text box and choose the DATAform command “Export all”. This text with DATAformTags will be exported:

```
DfXT+2.0¶*T107*e["Project1";841.89;595.276;55.555;56.666;57.777;58.888;1;3;9.999;0]*
$¶*#0*G0*T3*p1*@1*x57.777*y55.555*X536.388*Y785.224*W0*S0*C3*c9.999*i0*I0*
j0*J0*f0*H0*h0*D0*d0*B"White"*b1*F1*1"Helvetica"*20*312*41*5"Black"*61*70*80*
90*L128*l0*M"Black"*m1*R1*r1*w0*s0*|0*»0*«0*§0*N0*-1*+"1"*g"222222"*$Hello¶
```

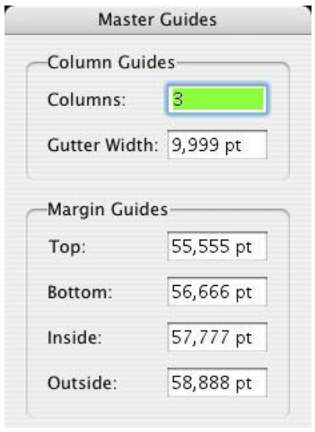
After *e[– or in the 4D interface with DX_Read(1) in DX_Array {69} – you receive eleven values, separated by semi-colons (the decimal point may in general be a point or comma in DATAformTags):

"Document1"	Name of the QuarkXPress project; resp. full path if saved already
841.89	Height of the layout in pt 841.89 / 72 = 11.693" see above illustration 841.89 / 2.834646 = 297 mm, i.e. DIN A4 letter height
595.276	Width of layout in pt 595.276 / 72 = 8.268" see above illustration 595.276 / 2.834646 = 210 mm, i.e. DIN A4 letter width
55.555	Margin guide top in pt
56.666	Margin guide bottom in pt
57.777	Margin guide inside/left in pt
58.888	Margin guide outside/right in pt

1	Facing pages is ON (0 for OFF)
3	Number of columns
9.999	Gutter width in pt
0	Automatic text box is always 0 with T107.

Document and master pages

A part of the document parameter is also shown as master guides.


Master Guides

Column Guides
Columns: 3
Gutter Width: 9,999 pt

Margin Guides
Top: 55,555 pt
Bottom: 56,666 pt
Inside: 57,777 pt
Outside: 58,888 pt

In a new document the values of the first master page are consistent with the values of the document. If you change the values of the master guides or create new masterpages, the documents values are not changed. The functional object *T107 always provides the original document values as they were created. These values cannot be changed at a later date in QuarkXPress.

Setup Document

Certain document parameters can be changed with the command "Document Setup…" from the QuarkXPress File menu, resp. by the comand File/Page Setup.
The functional object *T107 delivers these actual values of the document.

Import from T107

A *T107 object cannot be imported DATAformXTension will skip it. It is purely an information object for interpretation by a database.

But if the values *e[…] are transferred to the object *T108, the same document will be created again by DATAformXTension. (See under *T108, page: 104.)

*T108 CREATE DOCUMENT =>

¶*T108*e[…]*$¶

The functional object creates another new QuarkXPress document. (At least one document must already be open.)
The new document becomes the current QuarkXPress document; the import procedure is then continued in the new document.

If the following text with DATAformTags is imported:

¶*T3*$hello OLD¶*T108*e["new";841.89;595.276;55.555;56.666;57.777;58.888;1;3;9.999;0]*$¶*T3*p3*$hello NEW¶

a text box appears in the open document with the text “hello OLD”, then a new document named “new” is created and in this document, on page three, a text box is placed with the text “hello NEW”.

The properties of the new document are transferred within *e[…] in eleven parameters. (See under *T107, page: 102.)

All values are given in points.

The values received with *T107 create under *T108 the same document again. If the last, the eleventh parameter is 1, a document with automatic text boxes is created; otherwise 0 (zero) should be transferred.

Calling in the DATAform 4D interface

```
DX_Write (0)              `initialisation

        `Here may follow box objects for the “old” document

DX_Array{5}:="108"        `*T108 = create a new document

DX_Array{69}:="["+char(34)+"new"+char(34)+";841.89;
        595.276;55.555;56.666;57.777;58.888;1;3;9.999;0]"
        `the properties of the new document

$0:=DX_Write (1)          `transfer the object to the interface

        `Here follow the box objects for the new document

DX_Write (2)              `end of export
```

*T109 SAVE DOCUMENT =>

T109$¶ – without path details

Saves a QuarkXPress document that already exists as a file and works the same as the Quark-XPress menu command “Save”. If the document is still new or has been converted from an old version, the save dialogue appears.

*T109*$path¶ – with path details

¶*T109*$HD:my folder:my document¶
¶*T109*$C:\MyFolder\mydocument.qxd¶

Saves the document to the specified path. Files with the same name are overwritten. Corresponds to the QuarkXPress command “Save as...”, however no dialogues appear. The document itself cannot be overwritten. To save existing documents choose the above variant.
The path must correspond to the current operating system, the folders etc. must already exist.

Example under MacOS

```
¶*T108*e["NEW";841.89;595.276;55.555;56.666;57.777;58.888;1;3;9.999;0]*$¶*
T3*p1*$text box 1¶*T109*$HD:folder1:save 1.qxp¶*
T3*p2*$text box 2¶*T3*p3*$text box 3¶*T109*$¶
```

Example under windows

```
¶*T108*e["NEW";841,89;595,276;55,555;56,666;57,777;58,888;1;3;9,999;0]*$¶*
T3*p1*$text box 1¶*T109*$F:\folder1\save 1.qxp¶*
T3*p2*$text box 2¶*T3*p3*$text box 3¶*T109*$¶
```

Creates a new document called “NEW” and there on page 1 a text box;
saves the document as “save 1.qxp”;
creates a text box on page 2;
creates a third text box;
saves the document.

Calling in the DATAform 4d interface

```
DX_Write (0)              `initialisation

DX_Array{1}:="HD G3:folder1:save 1"
DX_Array{5}:="109"        `*T109 = save/save as

$0:=DX_Write (1)          `transfer object to interface

      `further objects may follow here

DX_Write (2)              `end of export
```

T110 SAVE AS EPS FILE =>, up to DATAformXTension 6.5 only*

*T110*e[...]*$

The functional object allows you to save a page or a part of a page as an eps file to disk.

Parameters

*e[PageNumber ; UseWholeSpread ; left ; top ; right ; bottom ; Scaling ; Flags]
*$Path to the eps file to be created

- Page number, or page number within the spread if UseWholeSpread. If the PageNumber is greater than the number of pages in the document, the last page will be used.
- UseWholeSpread = 1 for Save the whole spread to which the page belongs; = 0 for Save the specified page only
- Rectangle left ; top ; right ; bottom expressed in points
  - if UseWholeSpread = 1 this rectangle will be ignored
  - if UseWholeSpread = 0 and all 4 rectangle values are 0 => the whole page will be saved
  - if UseWholeSpread = 0 and the rectangle is not emtpy => this rectangle only will be saved as eps file.
- Scaling of the produced picture, e.g. 1 for 100%; 0.5 for 50% ; 1.35 for 135%. This referrs to the scaling of the picture itself, not to its placement by *Kk {52-53}.
- Flags, see the list below. Several flags can be added.
- Path to the eps file to be created, folders must be valid, path must be in OS convention. If a file name only is transferred, the file will be saved in the QuarkXPress folder.

Example

¶*T110*e[4;0;0;0;0;0;1;3]*$Test.eps¶
Save page number 4 in 100% as a colour EPS in PC format into the file "Test.eps".
Last paramter is 3, which means (EPSINCOLOUR = 1) + (EPSPCFORMAT = 2).

Example in the DATAform 4D interface

```
DX_write (0)                          `initialisation, start of export

DX_Array{5}:="110"                    `*T110 = Save as EPS file
DX_Array{69}:="[4;0;0;0;0;0;1;3]"     `the properties of the eps
DX_Array{1}:="Test.eps"               `name or path of the file
$0:=DX_Write (1)                      `transfer the object to the interface

      `Here may follow further objects

DX_Write (2)                          `end of export
```

Flags

EPSINCOLOUR = 1
EPSDODCS = 32
EPSDODCS2 = 64
These flags implement the choices in the "**Format**" drop-down of the "Save Page as EPS..." dialog. If none of these flags are used, a Black & White EPS file will be generated.

EPSNOPREVIEW = 1024
EPSPCFORMAT = 2 = preview in TIFF format
These flags implement the "**Preview**" drop-down of the "Save Page as EPS..." dialog.
If neither of these flags is used, then (on the Mac only) a PICT preview will be generated.
Add EPSPCFORMAT for cross platform usage of the picture.

EPS8BITDATA = 4096
EPSASCIIDATA = 16
These flags implement the "**Data**" drop-down of the "Save Page as EPS..." dialog, and have to do with how binary data is encoded in the output file. They specify that binary data will be encoded in "clean 8-bit" form or as regular ASCII characters respectively, using encoding algorithms. When the file is opened by a PostScript-aware device, that device will then decode it using the appropriate algorithms. Use only one of these flags at a time; if neither of them is used, a binary EPS file will be created.

INCLUDEIMAGEDATA = 0
OMITTIFF = 4
OMITTIFFANDEPS = 8
These flags implement the "**OPI**" drop-down of the "Save Page as EPS..." dialog. Use only one of these flags at a time.

Examples

You have to redirect the examples file paths to your volume and for Windows into OS convention – from HDG4:Test.eps into something like C:\Test.eps – if you don't use the QuarkXPress folder.

*e[PageNumber ; UseWholeSpread ; left ; top ; right ; bottom ; Scaling ; Flags]

```
¶*T3*p1*$Hello¶*
T110*e[1;0;0;0;0;0;1;3]*$Test.eps¶
```

- Creates a text box on page 1 with the text "Hello".
- Saves the page as the eps file "Test.eps", in the QuarkXPress folder, because a file name only is specified.

```
¶*T3*B2*-0*p3*x100*y200*X300*Y450*$Hello¶*
T110*e[3;0;100;200;300;450;1;3]*$Test.eps¶*
T12*p4*$Test.eps¶
```

- Creates a red text box at 100/200 200pt wide and 250pt high on page 3
- Saves this part of page 3 only as an eps file.
- Loads the eps file on page 4.

Use a QuarkXPress document with facing pages with this example:

```
¶*T3*B2*-0*p2*x100*y200*X300*Y450*$This is a red box¶*
T3*B4*-0*p3*x100*y200*X300*Y450*$This is a blue box¶*
T110*e[2;1;0;0;0;0;0,55;3]*$Test.eps¶*
T110*e[3;0;100;200;300;450;1;3]*$Testblue.eps¶*
T12*x0*X550*y0*Y300*p5*Z4*$Test.eps¶*
T12*p4*Z4*$Testblue.eps¶*
T105*$¶
```

- Creates a red box on page 2.
- Creates a blue box on page 3.
- Saves the entire spread of page 2 as an eps file - the picture includes page 2 and 3; the scaling of the picture will be 55%.
- Saves the blue box only of page 3 in "Testblue.eps"
- Loads the eps files on pages 4 and 5 (fitting the boxes to the pictures by *Z4).
- Shows the whole spread by calling *T105, fit in window.

Known restrictions

Binary EPS images
If a page contains binary EPS images a dialog will appear . The dialog text will say "Page contains EPS pictures which include Binary data. OK to continue?" or "Page could contain EPS pictures which include Binary data. OK to continue?".
There are two possible ways avoiding the dialogue:
- don' t generate eps files from documents containing binary eps images; use other eps types.
- create your eps file in binary format; use neither EPS8BITDATA nor EPSASCIIDATA.

Hint: Acrobat pdf

With Adobes Acrobat Destiller you can convert and combine eps files to pdf files.

IV. Message system

DATAformXTension contains functions for the implementation of a message system between the database and QuarkXPress. QuarkXPress may therefore be more closely linked to the database.

Advantage of the message system

- The steps "Export command + click in QuarkXPress window + import or update command" may be replaced with a single click. QuarkXPress remains in the background and displays the result.
- Box calculations may run in the background; QuarkXPress may also run in the background.
- With a single click all boxes of the current QuarkXPress document can be imported back into the database etc.

Application example

Three examples that are possible with DATAformXTension:

➡ The user clicks on the button "place in QuarkXPress" in his database: One or more article modules appear automatically on the QuarkXPress page. QuarkXPress remains in the background and displays the picture, the text box or several new pages with text and pictures.

⬍ A database needs the exact box heights of text boxes for layout calculation. It sends the text with DATAformTags with the message to QuarkXPress and receives back a new text with DATAformTags with the adjusted text boxes.

⬅ The user clicks on the button "re-update" in the database: The article or articles are exported from the QuarkXPress document as text with DATAformTags and the records in the database will be updated.

All three examples are implemented in the DATAform database. You may also try these examples in the demo version of the DATAform database and therefore get an idea of what is possible with regard to your requirements.

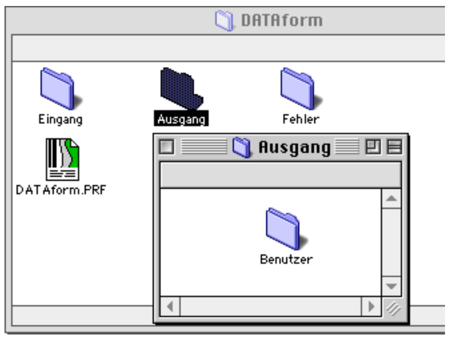
These functions turn QuarkXPress into a kind of QuarkXPress server to which tasks may be sent and from which you receive answers back.

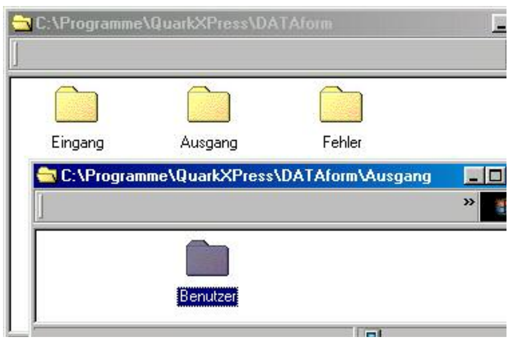
How it works

If the server is active, the DATAformXTension monitors the input folder *Eingang* for an incoming message file, processes the data, deletes the file and creates an answer file for the database in the folder *Ausgang/Benutzer*.

The database itself after having sent the message waits for the answer file from QuarkXPress in the folder *Ausgang/Benutzer*, reads the file and erases it.

The folders are created automatically by DATAformXTension when starting QuarkXPress:




C:\Programme\QuarkXPress\DATAform
Eingang
Ausgang
Fehler
C:\Programme\QuarkXPress\DATAform\Ausgang
Benutzer

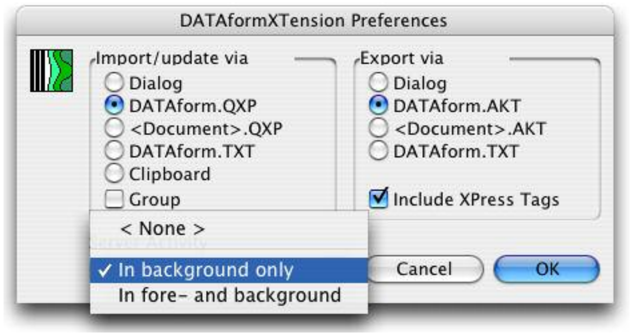
The input folder is the folder *[QuarkXPressFolder]/DATAform/Eingang*
The output folder is the folder *[QuarkXPressFolder]/DATAform/Ausgang/Benutzer*
Erroneous messages are placed in the folder *[QuarkXPressFolder]/DATAform/Fehler*

The *Eingang* and *Benutzer* folders are empty after a successful execution. If one of the partners could not collect a message, it will remain in the folder. Old messages may hinder the exchange and should be erased from all three folders. The DATAform 4D interface 3.0 or higher performs this automatically.

The folder names *DATAform*, *Eingang* etc. are language independent, they remain unchanged in an Italian or French QuarkXPress version. The database must therefore only know the path to the QuarkXPress folder and can derive all other paths from there.

Activate message system

Activate the server in QuarkXPress/DATAform/preferences...


DATAformXTension Preferences

Import/update via
- Dialog
- [✓ DATAform.QXP]
- <Document>.QXP
- DATAform.TXT
- Clipboard
- Group

Export via
- Dialog
- [✓ DATAform.AKT]
- <Document>.AKT
- DATAform.TXT
- [✓ Include XPress Tags]

Server Activity
- < None >
- [✓ In background only]
- In fore- and background

Cancel OK

< None >
The server is deactivated.

In background only
The server activity switches on if QuarkXPress is not the front most application or is hidden. If QuarkXPress is brought to the foreground, an active server procedure is completed; then the server activity is switched off. Messages which from this moment on are put in the *Eingang* folder are not processed. If QuarkXPress is once again sent to the background, the stacked messages will be processed one by one.
This is the default setting for a system in which the user works with QuarkXPress and the database.

Fore- and background.
The server activity is always active. Even while a user is working in QuarkXPress incoming messages will still be processed by QuarkXPress. The thought behind this option is using a QuarkXPress server on an individual computer in a network. With this setting a separate DATAform workspace could transfer tasks to this QuarkXPress computer.

The message system is platform independent. MacOS and Windows QuarkXPress versions may be used together.

Building a message

Message files are simple text files consisting of key value pairs separated by carriage returns. The file names can be chosen freely, but should be unique to avoid duplicates. The DATAform database builds the names as “seconds since midnight” + “txt”, e.g. “48788.txt”

An example of a message file to import a text with DATAformTags:

FROM: Benutzer
TO: DATAform
SUBJECT: IMPORT
DFTEXT: HD internal:DATAformImport.txt
DOCUMENT:

FROM:	The name of the folder in the folder *Ausgang* into which DATAformXTension will place its answer.
TO:	Constant *DATAform*
SUBJECT:	The command that DATAformXTension should execute. The following values are possible: IMPORT, EXPORTALL, IMPORTEXPORTALL, IMPORTEX-PORT, EXPORTGROUP.
DFTEXT:	The platform specific path to the text with DATAformTags file that should be imported or created upon export, such as C:\Catalogue\import.txt. The import settings of the preferences dialogue, import via clipboard, via DATAform.QXP etc. are not applied. If you wish to use the file *DATAform.QXP* upon import in the message system, you transfer under DFTEXT the path: *[QuarkXPress folder]:DATAform: DATAform.QXP*. The DATAform database also uses this method. It has the advantage that the import file may be, for example, imported from the menu command “Import boxes” even if the server is deactivated.
!	For an import via message the database creates two files in any case: the message file and the DATAformTags file that should be imported.
DOCUMENT:	A path to a QuarkXPress document may be specified here. The document will then be opened first. If the path is missing, the command relates to the currently opened document.

After processing the command DATAformXTension creates a file with the same name in the folder *[QuarkXPressFolder]:DATAform:Ausgang:Benutzer* with i.e. the following content:

FROM: DATAform
TO: Benutzer
SUBJECT: IMPORT
DFTEXT: HD internal:DATAformImport.txt
DOCUMENT:
RESULT: 0, error number: 0

If an error appears, it will be transferred after RESULT with a text message: e.g.:

RESULT: 65584, There is no current QuarkXPress document
RESULT: 65493, The file could not be found. (-43)

The text message is language dependent. In a German QuarkXPress it is German otherwise English. A list of all error messages may be found under error messages in the appendix.

If the message file cannot be processed, because e.g. after SUBJECT: no valid command is given, the file will be moved to the error folder *DATAform:Fehler*.

If you use the DATAform 4D interface, the command DX_Message deals with the message files instead: DX_Message creates the message, waits for the answer and evaluates it. (See under DATAform 4D interface.)
The procedures of the 4D interface are – also as a text file – part of the DATAform developers kit and may be used as a coding template in other systems.

Messages and functional objects

The transferred text with DATAformTags may also contain functional objects.

DATAformTag Text without functional objects

- The DATAform setting "Group" is used upon import if the check box is set to ON.
- Existing boxes within the document with the same object number will be skipped upon import; the duplicate dialogue with the message "the box number already exists " will not appear.
- Likewise if boxes are not found when updating the content, they will be skipped without a message.
- If a picture is not found, the empty picture box is created. The load picture dialogue will not appear.

Example: The file *DATAformImport.txt* contains this text with DATAformTags, "Group" in the preferences is OFF.

¶*T3*G1*#1*$Text¶*x110*X210*T12*G1*#2*$False picture path¶

One text and one picture box are imported, no load picture dialogue appears, the boxes will not be grouped.

DATAformTag Text with functional objects

The settings in the preferences dialogue can be overwritten via functional objects:

¶*T106*M1*$¶*T101*M1*$¶*T3*G1*#1*$Text¶*x110*X210*T12*G1*#2*$False picture path¶*x220*X320*T12*G1*#3*$False picture path 2¶

- The boxes will now always be grouped by *T106*M1. By *T101*M1 the load picture dialogue will appear with non found pictures.

All functional objects may be part of the text with DATAformTags even if it is imported by the message system.

Functional objects here also have priority over the command SUBJECT:

Example: The message contains SUBJECT:IMPORT
The DATAformTag however, reads:

¶*T102*M1*$¶*T3*G1*#1*$Text¶

*T102 switches the processing mode to *update*, the following boxes are therefore not imported, but DATAformXTension tries to update existing boxes.

IMPORT

FROM: Benutzer
TO: DATAform
SUBJECT: IMPORT
DFTEXT: HD internal:DATAformImport.txt
DOCUMENT:

This command imports the text with DATAformTags located at the path *DFTEXT*: into the QuarkXPress document which is given in the fifth argument. The command is equivalent to “Import boxes” in the DATAform menu in QuarkXPress.

Without functional objects existing boxes in the document with the same object number are skipped.

The Tag *{60} *Δ Adjust box height* => will be processed, see the example under *IMPORTEX-PORT*.
At the end the boxes will be grouped if *grouping* is set to ON in the preferences or the functional object *¶*T106*M1*$¶* was transferred.

UPDATE

FROM: Benutzer
TO: DATAform
SUBJECT: UPDATE
DFTEXT: HD internal:DATAformImport.txt
DOCUMENT:

This command updates the objects of the QuarkXPress document via the text with DATAform-Tags. It is equivalent to the command “Update contents” in the DATAform menu in Quark-XPress.

Without functional objects non found boxes will be skipped.

The Tag *{60} *Δ Adjust box height =>* will be processed. At the end the boxes will be grouped according to the preferences.

EXPORTALL

FROM: Benutzer
TO: DATAform
SUBJECT: EXPORTALL
DFTEXT: HD internal:DF Export
DOCUMENT:

This command exports all objects of the current QuarkXPress document into the DATAform-Tags file at the given path. The DATAformTags file will be created at this path. A file with the same name will be overwritten.
It is equivalent to the command "Export all" in the DATAform menu. Boxes on master pages are not exported.

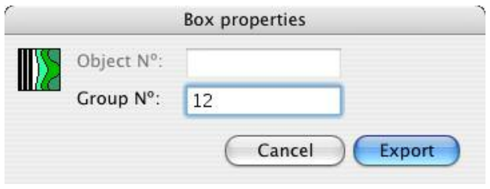
The DATAformXTension setting "With XPressTags" will be applied if the checkbox is ON.

EXPORTGROUP

FROM: Benutzer
TO: DATAform
SUBJECT: EXPORTGROUP
PARAMETER: 12
DFTEXT:XPRESS:DATAform:DATAform.AKT
DOCUMENT:

This command exports all DATAform boxes with a certain group number. The group number is specified under PARAMETER:.

The command is equivalent to the menu command DATAform/Export group. E.g. all boxes in the document with the group number 12 will be exported with these settings:



Box properties
Object N°:
Group N°: 12

DATAform 4D interface

In DX_Message there is no special argument for *PARAMETER*: The calling of *EXPORT-GROUP* is passed as a second line of the first argument:

```
$command:="EXPORTGROUP" + Char( 13 )
$command:= $command + "PARAMETER:"+ String( Group number )
$err:=DX_Message($command;$DFTEXT;"";$XPFolder;Timeout; =>T1)
```

In this context “Group number” is the group number of the DATAform boxes that should be exported.

The command DX_Message hence creates a message file as shown above, waits for an answer and evaluates it.

IMPORTEXPORTALL

FROM: Benutzer
TO: DATAform
SUBJECT: IMPORTEXPORTALL
DFTEXT: HD internal:DATAformImport.txt
DOCUMENT:

This command is a combination of *IMPORT* and *EXPORTALL* and is performed in several steps:

- The text with DATAformTags is imported into QuarkXPress, cf. *IMPORT*. The Tags *{60} *Δ Adjust box height =>* will be processed, see the example under *IMPORTEXPORT*.
- Then all document boxes, including those without DATAform box ID, are exported cf. *EXPORTALL*. The specified DATAformTag file is overwritten.
- After this the imported objects, not all objects of the document, are deleted once again. Pages newly created when importing are not deleted. Finally the remaining boxes are grouped according to the preferences.

Example: The imported DATAformTag file *DATAformImport.txt* contains at first only the box:
¶*T3*$The first box¶
The QuarkXPress document already contains a box with the text “existing box”. At the end of the command the file *DATAformImport.txt* contains the document parameters and the complete text with DATAformTags of both boxes.

```
DfXT+2.0¶*T107*e["Document1";841.89;595.276;55.555;56.666;57.777;58.888;1;3;9.999;0]
*$¶*#0*G0*T3*p1*@1*x57.777*y55.555*X536.388*Y785.224*W0*S0*C3*c9.999*i1*I1*
j1*J1*f0*H0*h0*D0*d0*B"White"*b1*F1*1"Helvetica"*20*312*41*5"Black"*61*70*80*
90*L128*l0*M"Black"*m1*R0*r1*|0*»0*«0*§0*N0*-0*+"1"*$existing
box¶*#0*G0*T3*p1*@1*x0*y0*X100*Y100*W0*S0*C1*c12*i1*I1*j1*J1*f0*H0*h0*D0*
d0*B"White"*b1*F1*1"Helvetica"*20*312*41*5"Black"*61*70*80*90*L128*l0*M"Black"*
m1*R0*r1*|0*»0*«0*§0*N0*-0*+"1"*$The first box¶
```

Cf. also *IMPORTEXPORT*

IMPORTEXPORT

FROM: Benutzer
TO: DATAform
SUBJECT: IMPORTEXPORT
DFTEXT: HD G3:DATAformImport.txt
DOCUMENT:

This command imports the boxes of the specified DATAformTags file, exports these boxes again and deletes them. The boxes that are already present in the QuarkXPress document are not exported. Newly created pages are not deleted again.

This command serves mainly for calculating the box height: The Tag *{60} *Δ Adjust box height* => will be processed, then the boxes with the new *Y values are once again exported and deleted in the QuarkXPress document. The boxes already present in the QuarkXPress document are left unchanged.

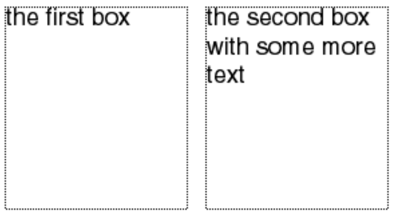
Example: The imported DATAformTag file *DATAformImport.txt* contains the boxes:

¶*x10*X100*y20***Y120***T3*Δ1*$the first box¶*x110*X200*y20***Y120***T3*Δ1*
$the second box with some more text¶

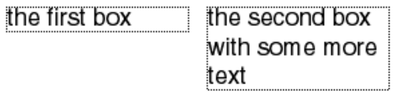
Both box descriptions contain the DATAformTag *Δ1. At the end of the command the file *DATAformImport.txt* contains the complete text with DATAformTags of both boxes:

DfXT+2.0¶*#0*G0*T3*p1*@1*x10*y20*X100***Y33.509***W0*S0*C1*c12*i1*I1*j1*J1*f0*H0*h0*D0*d0*B"White"*b1*F1*1"Helvetica"*20*312*41*5"Black"*61*70*80*90*L128*l0*M"Black"*m1*R0*r1*I0*»0*«0*§0*N0*-0*+"1"*$the first box¶*#0*G0*T3*p1*@1*x110*y20*X200***Y62.178***W0*S0*C1*c12*i1*I1*j1*J1*f0*H0*h0*D0*d0*B"White"*b1*F1*1"Helvetica"*20*312*41*5"Black"*61*70*80*90*L128*l0*M"Black"*m1*R0*r1*I0*»0*«0*§0*N0*-0*+"1"*$the second box with some more text¶

- Both boxes are imported and receive a box height of *Y120-*y20 = 100 pt



- Then the box heights are adjusted to the text quantity:



- The boxes are exported as a text with DATAformTags with the new *Y values and are then deleted from the QuarkXPress document.
  The text with DATAformTags supplies the new heights:
  ***Y33.509**-*y20 = 13.509 pt for the first box and
  ***Y62.178**-*y20 = 42.178 pt for the second one.

CLOSE

FROM: Benutzer
TO: DATAFORM
SUBJECT: CLOSE
DFTEXT: 1

The command closes the current (top level) QuarkXPress document.

DFTEXT: 0 Closes the document without saving it.
DFTEXT: 1 Saves the document and closes it.
Only documents, which have been saved once, can be saved.
Newly created documents cannot be saved with this command.
(For saving documents cf. the functional object *T109.)

If no error occurs DATAformXTension responds with:

FROM: DATAFORM
TO: Benutzer
SUBJECT: CLOSE
DFTEXT: 0
DOCUMENT:
RESULT: 0, Error number: 0

CLOSE message in the DATAform 4D interface
The CLOSE command is transferred to DX_Message as follows:

```
$err:=DX_Message ($Cmd;$Save;$XPDoc;$XPFolder;$Timeout;=>T1)
```

$Cmd	"CLOSE"
$Save	"0" = do not save; "1" = save
$XPDoc	Is always ""; no QuarkXPress document file is called here.
$XPFolder	The valid path to the QuarkXPress folder
$Timeout	Timeout e.g. after 3 seconds of waiting time
T1	Timeout text and variable for the response error text.

Example close without save:
```
$err:=DX_Message ("CLOSE";"0";"";$XPFolder;3;=>T1)
```

Example close with save:
```
$err:=DX_Message ("CLOSE";"1";"";$XPFolder;3;=>T1)
```

DX_Message creates a message file in the folder DATAform/Eingang, waits for the response of DATAformXTension and processes it.
If the document was closed, $err will become Zero.

PRINT, *up to DATAformXTension 6.5 only*

FROM: Benutzer
TO: DATAFORM
SUBJECT: PRINT
DFTEXT: 2 4 1

The command prints the current (usualy top level) QuarkXPress document.
The command supports the definition of the page range and the number of copies.
All other preferences are taken as set in the printing preferences of the document.

DFTEXT: 2 4 1 means:
fromPage toPage Copies, separated by spaces.
If toPage is higher than the number of pages in the document, all pages until the end of the document are printed.

DFTEXT: 1 9999 1
Prints all pages of a document.

If no error occurs DATAformXTension responds with:

FROM: DATAFORM
TO: Benutzer
SUBJECT: PRINT
DFTEXT: 2 4 1
DOCUMENT:
RESULT: 0, error number: 0

PRINT message in the DATAform 4D inteface

The PRINTcommand is transferred to DX_Message as follows:

$err:=DX_Message ($Cmd;$Pages;$XPDoc;$XPFolder;$Timeout;=>T1)

$Cmd	"PRINT"
$Pages	"2 4 1" `fromPage toPage Copies
$XPDoc	Is always ""; no QuarkXPress document file is called here.
$XPFolder	The valid path to the QuarkXPress folder
$Timeout	Timeout e.g. after 3 seconds of waiting time
T1	Timeout text and variable for the response error text.

Example: Print alle pages of the document:

$err:=DX_Message ("PRINT";"1 9999 1";"";$XPFolder;3;=>T1)

DX_Message creates a message file in the folder DATAform/Eingang, waits for the response of DATAformXTension and processes it.
If the document was sent to the printer or spooler succesfully, $err will become Zero.

V. DATAform database interface

This interface offers a ready to use connection between a 4D database (www.4D.com) and DATAformXTension for QuarkXPress.

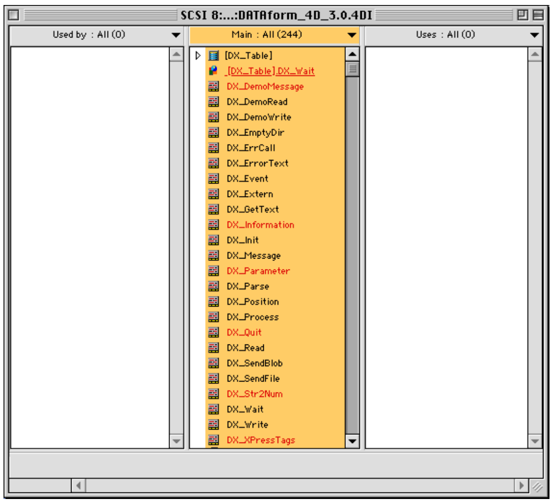
If you use another database language than 4D, you must develop similar procedures in your language. The document “DATAform 4D...TXT” contains all the procedures as plain text. Take these procedures as a suggestion or template.

The interface simplifies and standardises the reading and writing of a text with DATAform-Tags. It is basically a translator of DATAformTags into the text array DX_Array and likewise creates a text with DATAformTags from the DX_Array. The user of the DATAform 4D interface therefore writes to and reads from the DX_Array only. He has no direct contact with the text with DATAformTags, but uses just the Array.

In addition to this the interface contains with *DX_Message* the complete handling of the message interface for the remote control of QuarkXPress.

Installation

The procedures are available as a 4D insider document and as a text file.
The 4D insider document contains:



```
[DX_Table]
_[DX_Table].DX_Wait
DX_DemoMessage
DX_DemoRead
DX_DemoWrite
DX_EmptyDir
DX_ErrCall
DX_ErrorText
DX_Event
DX_Extern
DX_GetText
DX_Information
DX_Init
DX_Message
DX_Parameter
DX_Parse
DX_Position
DX_Process
DX_Quit
DX_Read
DX_SendBlob
DX_SendFile
DX_Str2Num
DX_Wait
DX_Write
DX_XPressTags
```

Copy all the components from this window into your own 4D structure:
- Select all components (command A) and drag them to your application.
- With a later update installation set 4D insiders Moving Preferences for forms and methods to “Replace”. After moving all elements 4D insider will state: “Tables [DX_Table]: This table already exists.” Choose the option “Do not copy”. OK. Done.

The procedures are conceived for use in compiled form; the database should be compiled with 4D compiler.

The 4D method DX_Information shows an overview of the interface.

DX_Information

```
` DATAformXTension 4D interface
` The interface simplifies and standardises the writing
` and reading of DATAformTags
`---------------------------------------------------
`    DATAformXTension interface for 4th DIMENSION
`    GASSENHUBER Systementwicklung, Germany
`    www.gassenhuber.de
`---------------------------------------------------
` Version 02.05.2002, {73} *?, {74} *=[...], {75} *g, DX_DirSep
` Version 04.09.2000, DX_Str2Num, DX_GetText
` Version 09.02.2000, DX_SendBlob
` Version 06.07.1998 for 4D V6


`____________________________ COMPONENTS ____________________________

` In total the interface contains 25 methods and 1 form in all.

`          These 5 “Methods” provide information and examples.
` DX_Information     This information.
` DX_Parameter       Contains a list of all parameters.
` DX_DemoWrite       Demonstrates the transfer of objects to QuarkXPress.
` DX_DemoRead        Demonstrates the reading of objects from QuarkXPress.
` DX_DemoMessage  Demonstrates the usage of the message interface.

`        These 3 methods compose the main interface.
`# DX_Write
`# DX_Read
`# DX_Message

`        These 14 methods are called internally by the core methods.
`# DX_Position
`# DX_ErrCall
`# DX_Event
`# DX_Extern
`# DX_Init              DX_Init (-1) must be called in your startup method.
`# DX_Parse
`# DX_Process
`# DX_SendBlob
`# DX_SendFile
`# DX_Wait
`# DX_GetText
`# DX_Str2Num
`# DX_EmptyDir
`# DX_DirSep

`        These 3 methods are called by you.
`# DX_Quit              Call DX_Quit in your On Exit method before QUIT 4D.
` DX_ErrorText       Shows the last error: ALERT ( DX_ErrorText )
` DX_XPressTags     This function may be used by you.
`                          It separates the XPressTags header from text.
```

```
`# The form DX_Wait in [DX_Table] with its form method,
` cf. the method DX_Wait


`____________________________ INSTALLATION ____________________________

`1- Import all contents, at least those marked with #.
`     Don't forget to copy the form DX_Wait in [DX_Table].

`2- In your startup method you write: DX_Init (-1)
`     In the line before you define the language for all DATAform alerts:
`     ◊DX_Language:=2   `2 = English or
`     ◊DX_Language:=1   `1 = deutsch

`3- In your On Exit method you call DX_Quit before QUIT 4D
`    if you work with DX_Message.

`4- Restart 4D. Done.

` Now you may write a method similar to DX_DemoWrite and DX_DemoRead
` and exchange QuarkXPress objects with your data file.


`____________________________ LIST OF VARIABLES ________________________

` Used and initialised variables are:
` See DX_Init (-1)

` The function DX_Message calls at the end
` ON ERR CALL("") as well as ON EVENT CALL("").
` If your programme calls its own methods here,
` you must reset them after calling DX_Message.


`____________________________ MAC/WINDOWS ____________________________

` The DATAform 4D interface can be used under 4D for Mac or Windows.
` The interface automatically performs the converting of the Mac ANSI
` character set upon import and export.
` You may import at will text with DATAformTags from QuarkXPress for Mac or Windo
` into Mac or Windows as long as it has been created with version 3.7 or later
` of DATAformXTension.
` You may create text with DATAformTags under Mac or Windows and import it
` with either OS in QuarkXPress.
` If necessary, DATAformXTension converts the character set automatically.

` DX_Message expects a path to the QuarkXPress folder.
` This path to can be passed in Mac or Windows notation.

` If you pass a picture path in DX_Array{1}, you must however check
` yourself that the path is suitable to target OS.
```

The three methods of the core interface

DX_Write

DX_Write (Selector; {path}) => Boolean

The DX_Write function serves to transfer objects to QuarkXPress. The transfer can be done via the clipboard or via a file. When importing into QuarkXPress, the DATAformXTension has to be set to import from the clipboard or from a file. (See under DATAformXTension “Preferences...”.)

The transfer of objects follows, in general, this coding scheme:

```
DX_Write (0;{path})         `initialise the interface
Loop through the export records
        Fill the DX_Arrays with the values of a box
        DX_Write (1)        `Transfer the filled DX_Array to the interface
        Load next record
End of loop
DX_Write (2)                `Transfer all collected objects to QuarkXPress.
```

The DX_DemoWrite procedure shows the procedure.

The calls must occur in the order 0, 1, 2.
DX_Write returns True if no error occurred. The variable "DX_Error" becomes 0.
DX_Write returns False if an error occurred. The error number is then written into the variable DX_Error.

DX_Write (0;{path}) => Boolean
Initialises the text array DX_Array.
If a path is transferred, the file is created and opened. If the path is missing, the clipboard will be used.

DX_Write (1) => Boolean
Transfers the DX_Array (the object) to the interface. The interface translates the object into DATAformTags and places it in a buffer.
If a DX_Array line begins with “*”, the line is transferred as it is without translation, cf. DX_Read {3}.

DX_Array{1} and DX_Blob
The text/picture path of the box is usually transferred to DX_Array{1}.
With chained boxes the entire text of the chain must be transferred with the first box.
This may lead to large texts that extend over the 30K limit for DX_Array{1}. In this case the whole text should be transferred to the Blob DX_Blob.

Many texts are added as shown in the following scheme:

```
TEXT TO BLOB(Text1;DX_Blob;Text without length;*)
TEXT TO BLOB(Text2;DX_Blob;Text without length;*)
TEXT TO BLOB(Text3;DX_Blob;Text without length;*)
```

At the end a DX_Write {1} follows for the entire text of the box.

Per DX_Write(1) you can switch between the transfer to DX_Array{1} and DX_Blob. If the blob contains data, its data will be used and not the data in the DX_Array{1}. The blob will then be written into the file by DX_Write{1} and reset to zero.

If you work with DX_Blob, the data will not be collected anymore, but written straight away into the file. With small text quantities and picture paths it is better to fill the DX_Array{1} in order to reduce hard drive access.

DX_Write (2) => Boolean
Transfers all objects that were previously transferred to the interface to the clipboard or closes the file if a path is given. Frees up memory.

DX_Write (3) => Boolean
Transfers the DX_Array directly to the interface or to the file. {1} and {2} are done in one step, the DX_Array is not erased. Because objects are not collected, only one object can be transferred. DX_Write {3} is useful for test purposes only.

DX_Read

DX_Read (Selector; {path}) => Boolean

The reading of text with DATAformTags is structured as follows:

```
DX_Read (0;{path})          `Open updating file
While (DX_Read (1))         `Read object and fill the DX_Array
    Search for the corresponding record by the object No. in DX_Array{2}
    Update/create/skip the record
    Save record
End while                   `End of loop
```

The procedure DX_DemoRead demonstrates the procedure.

DX_Read (0;{path}) => Boolean

Starts a read procedure. If a path is specified, the file will be opened. The validity of the path should be checked before transfer. If the path is missing, the open file dialogue appears and you may choose an update file from there.
DX_Read returns True if an appropriate document has been opened, otherwise False and an error number in DX_Error.

DX_Read (1) => Boolean

Reads an object from a document that has been successfully opened by DX_Read {0}. The object will be read into the Array DX_Array.
DX_Read returns True if successfully read otherwise False and an error number in DX_Error.

DX_Read (1) is usually called repeatedly until all objects will have been read.
If the end of the document is reached, DX_Read {1} returns False, the document is closed, the DX_Array is erased, the variable DX_Error is set to 20004.

DX_Read (2) => True

Closes a document. This call is necessary only if the read procedure should be interrupted. It always returns True.

DX_Read (3) => Boolean

Reads an object like {1}, but only the first 10 plus a further 6 lines of the Array will be filled with the values. The remaining Tags are not read as values, but are placed as a text with DATAformTags in DX_Array{11}. With this method most of the object properties are received as a text and can be saved in a text field as a block of text.

With your following transfer to QuarkXPress you fill the line {11} with the complete text again. DX_Write {1} transfers the text unchanged as it starts with "*", cf. DX_Write {1}.

By DX_Read {3} the first 10 lines of the Array will be filled and additionally:

{28} *F Font mode; if line {28} = "2", the text includes XPressTags.
{63} *+ Section page <=, delivers the section page on which the box is placed.
{64} *» Chain: next box
{65} *« Chain: previous box
{66} *A Anchor in box
{71} *@ Character set table

The remaining exported Tags are written as DATAformTags text in the Array line DX_Array{11}.

DX_Read (4) => Boolean

Allows you to choose for every single DATAformTag whether it should be written into the DX_Array or added as text to DX_Array{11}. For this you write a string in DX_Array{0} composed by Char (desired DX_Array line).

DX_Read (4) only reads these DX_Array {0} parameters into the DX_Array. All others in DX_Array {11}.

Example for DX_Read (4)

```
DX_Array{0}:=Char(1)+Char(2)+Char(3)+Char(4)+Char(5)+Char(6)+Char(7)
DX_Array{0}:=DX_Array{0}+Char(8)+Char(9)+Char(10)+Char(28)+Char(63)
DX_Array{0}:=DX_Array{0}+Char(64)+Char(65)+Char(66)+Char(71)+Char(73)
```

If you fill DX_Array{0} with this string, DX_Read {4} behaves identically to DX_Read {3}: Only parameters 1-10, 28, 63-66 and 71, 73 will be read in their DX_Array line. All others in DX_Array {11}.

By adding or erasing a parameter in the list you can define individually whether the parameter should be transferred as an Array line or as text in DX_Array {11}.
In this way can e.g. also the anchoring information be saved in one text together with other box properties or in specific, separate fields.

DX_Buffer for DX_Read
The function DX_Read works optionally with an internal memory buffer.

The import procedure is much faster when this buffer is used (at least by a factor of 2.)
If the buffer is used, the maximum object size is 16k. The object size is the number of characters that are needed to describe a QuarkXPress box (including its text) meaning all characters between ¶ and ¶.
If the buffer is not used, the maximum object size is 32k.

Turning the buffer on and off
The buffer is by default switched on by the function DX_Init(-1), called up in your startup procedure.
To switch the buffer off you write after the line DX_Init(-1):

```
        DX_Buffer:=False    `Max. object size: 32K
```

The buffer can then be switched on in another place in your programme by

```
        DX_Buffer:=True    `Max. object size: 16K.
```

The variable DX_Buffer is checked by DX_Read and used in the next import procedure.

DX_FilePos for DX_Read
Shows the current position in the open file. Every read procedure by DX_Read moves the position. DX_FilePos can be used, for example, to show the progress of the read procedure:

The file length is read in a variable before the read procedure.

```
◊FileSize:=Get document size (path)
```

In the actual read loop the progress can then be shown as percentages, i.e.:

```
$i:=0
While (DX_Read (3))
   $i:=$i+1
   If ($i%10=0)
   MESSAGE("Objects: "+String($i)+(" "*10)+String(DX_FilePos/◊FileSize*100; "## %"))
   end if
End while
```

DX_Message

DX_Message (command; [DFTEXT]; [XPDOC]; [XPFolder]; Timeout; ‘Text; {user}; {form name }) => Error
DX_Message (Text; Text; Text; Text; int; TextPtr ; {Text}; {Text}) => Integer

This function sends a message to QuarkXPress and transfers the arguments.
DATAformXTension receives the message, performs it and sends a message back.
DX_Message waits for the response and returns the result as function value.

Example:
myText:= "QuarkXPress doesn’t respond."+char(13)+"Waiting:"
$XPFolder := "HDinternal:QuarkXPress folder:"
$err := DX_Message ("IMPORT" ; "HDinternal:DFMText ; "" ; $XPFolder ; 5 ; =>myText)

DATAformXTension will import the text with DATAformTags “DFMText” into the current QuarkXPress document. At the end it sends a response and confirms the process or shows any errors that have occurred.
The function DX_Message waits for up to 5 seconds (timeout as 5. argument) for the response.
If the response does not arrive within this time, it shows the second alert text myText.
If the response arrives, it shows the result otherwise –1 for timeout.
A text with DATAformTags is hence imported the same way as with the command “Import boxes” in the DATAform menu in QuarkXPress.

The function DX_Message expects between 6 and 8 arguments:

1) Command
The following values are possible: IMPORT, UPDATE, EXPORTALL, EXPORTGROUP, IMPORTEXPORTALL, IMPORTEXPORT, CLOSE, PRINT.
These commands and the internal functioning of the message system are described in detail in chapter *IV. Message system*, page: 109f.

2) Text with DATAformTags
In the second argument the complete path to a text with DATAformTags file is transferred. The text will be imported by the import functions. This file will on the other hand be created or overwritten by the export commands.

3) QuarkXPress document
In the third argument you specify the QuarkXPress document upon which all commands should act. The possibilities are:

Complete path
If a complete path is transferred, the QuarkXPress document will be opened.

Empty argument
If an empty string is specified, "", the commands relate to the current QuarkXPress document.

Only use one QuarkXPress document. Switching between different opened QuarkXPress documents is not supported.

4) QuarkXPress folder
In the fourth argument you specify the path to the QuarkXPress folder.

5) Timeout
In the fifth argument you specify the maximum waiting time in seconds. DX_Message waits for the specified time for the answer to arrive.
The wait loop can always be interrupted by command point.
If you specify 0 as the timeout, DX_Message doesn't wait or give the user the possibility to extend the wait period.
You have the possibility to install your own cancel dialogue and to create your own error values. (See the method DX_DemoMessage.)
Note also any further instructions in the header of the DX_Message method.

6) Text pointer
The text pointer of the sixth argument functions as an input as well as an output variable.
You specify a pointer on a global text variable, e.g.:
C_Text (myText)
And then *DX_Message* (…; =>myText ;…)

<u>Input</u>
If you transfer a text in myText, the dialogue shows this text in case of a timeout and the user may extend the wait time.
If myText is empty, the interface stops waiting at timeout and the time cannot be extended.

<u>Output</u>
At the end DX_Message transfers in myText:
- In a timeout case the name of the created message file.
- If the message is received, the error text that DATAformXTension delivered is transferred in myText.

For further details see the header of function DX_Message.

7) User
In the fourth argument you may eventually specify a user ID. This is only necessary if more than one user works with the same QuarkXPress programme.
By the user ID the answer can be linked to the specified user.

If only one user works with his QuarkXPress an empty string may be specified in the fourth argument or the argument may be left out. Then DATAformXTension places its answer in the *[QuarkXPress]:DATAform:Ausgang:Benutzer* folder. The DX_Message function looks for the answer at this location.

If several users work in a network on the same QuarkXPress programme, specify unique names for all users. For each of these users, create a folder with the same name in the *Ausgang* folder. Specify these names as the seventh argument. DX_Message instructs DATAformXTension to place the answer in this folder and waits there for the answer. The folder must already exist.

File names
DX_Message places all messages to XTension in the folder *[QuarkXPress]:DATAform:Eingang*. It automatically creates unique file names with "seconds after midnight". It then waits for a file with the same name in the folder *Benutzer*. It reads the file and erases it.

8) Form name
Here the name of a special wait form can be given. DX_Message will then use this form for the Cancel dialogue. The form must be present in the table [DX_Table].
If the eighth argument is missing, the form "DX_Wait" in [DX_Table] is used. The form must exist in [DX_Table] otherwise an error message is given. (See also under the method DX_Wait.)

Return values of the function
DX_Message returns the following values:

0 No error occurred; the DATAformXTension response is received and it contains the error value zero.

< 0 DX_Message generates the following negative error numbers:
Error -1 = timeout.
No message received. Possible causes:
The wait time was too short. QuarkXPress is not open. DATAformXTension is not loaded. The message system is deactivated; the server activity is set to "none". The demo time of an XTension has run out (20 Minutes from QuarkXPress start). The folder structure in the DATAform folder is incorrect. (See under message system.)
Error -2 = timeout, $5 was = 0, the answer was not waited for.
Error -3 = timeout, wait loop interrupted by user.
Error -4 = timeout, the confirm dialogue was canceled.
Error -10 = the command, parameter $1, is missing.
Error -11 = the path to the DATAformTags file $2 is missing.
Error -12 = the path to the QuarkXPress folder $4 is missing.
Error -13 = fewer than 6 arguments were transferred.
Error -20 = the message file could not be created.
Error -21 = the message could not be placed in the input folder *Eingang*.
Error -30 = the process could not be started.
Error -40 = a message was received, but could not be read.

>0 The response of the DATAformXTension was received and contains an error value >0. DATAformXTension simultaneously sends an error text with the response. The text is transferred in $6=>. A list of error messages from 65537 to 65590 may be found in the appendix.

How it works

Implementing the message system in your 4D application, demonstrating a "automatic box height calculation".

Create a new procedure, e.g. CalcBoxes.
CalcBoxes contains the following three steps:

1) Start QuarkXPress

Check manually or via a method:
- If QuarkXPress is open. If not start QuarkXPress.
- If the QuarkXPress document exists into which you wish to place.
Or start QuarkXPress manually with the document. Make sure that the settings in QuarkXPress "Include XPressTags" are appropriate for your needs. Turn the server activity in DATAformXTension to ON.

2) Create the DATAformTag file

MyExport `This is your export method, it creates e.g. the document
`„DATAform.QXP“ in the QuarkXPress:DATAform folder.

In this method you set the DATAformTag *DX_Array {60} *Δ* to the desired value, e.g.:
DX_Array {60} := "1"

Check the success of the export: The file DATAform.QXP was created.

3) Send a message to DATAformXTension

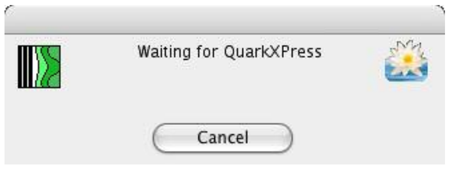
```
$err := DX_Message ( "IMPORTEXPORT" ; DFMText; XP-Doc ; XPƒ; 5; =>myText )
          ` DFMText is the path to the file created by MyExport.
          ` XP-Doc is the path to the QuarkXPress document and may be "".
          ` XPƒ is the path to your QuarkXPress folder.
          ` 5 Timeout; set the value relative to the import file size.
          ` myText now contains your timeout alert text. At the end it may contain an error text
          ` from DATAformXTension or the name of the created message file.
```

DX_Message creates a message file in the monitored input folder *Eingang*.

DATAformXTension receives the message, imports the file *DFMText*, adjusts the box heights depending on *Δ to the text and exports the boxes again. The file *DFMText* is overwritten by the exported new data.

Then DATAformXTension creates its answer file.

During this time *DX_Message* and your programme both wait.



DX_Message shows the wait dialogue. (The form "DX_Wait" or $8 in [DX_Table])

If the message is received, your programme continues executing from here. If the user clicks on Cancel in the wait dialogue, $err will become -3, see above.

```
if ($err # 0 )
  ShowError ($err)
                 `Your method processes the error and eventually indicates it etc.
else
  MyImport       `Import the file DFMText path as usual and process the
                 `received new box parameters.
End if
```

Note further details in the header of the methods, especially DX_Message and DX_DemoMessage.

If you are using 4D, see the DATAform 4D sample database, available for free at www.gassenhuber.de. It uses DATAform interface and is supplied in compiled and not compiled form.

VI. Appendix

Text with XPressTags

XPressTags describe the font and style properties of characters and the formatting of paragraphs within text boxes. XPressTags are to be used whenever the text in a box should have varying properties.

All paragraph properties may be transferred by style sheet calls, with the advantage of defining or modifying all properties by the style sheet in QuarkXPress. Varying properties within a paragraph can be transferred via character style sheets.

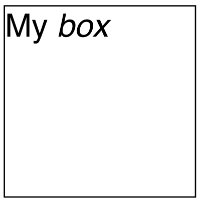
Spelling and functioning of XPressTags are described in detail
- in the QuarkXPress manuals and
- in the document *XPress Tags ReadMe.pdf* e.g. on DATAform CD.

To import or export text with XPressTags the XPressTags filter must be loaded.

Import

When importing text boxes XPressTags are applied if the Tag *F is set to *F2. (See under *{28} *F Font mode*, page: 51)

If the text contains XPressTags, these are part of the text.
¶*T3*F2*$My <PI>box¶
creates a text box and sets the word “box” to italic:


My *box*

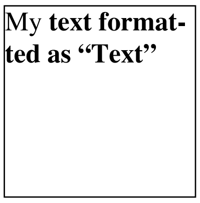
The same result can be achieved with a character style sheet:
You create a character style sheet, i.e. with the name and property “italic” and call it as below:

¶*T3*F2*$My <@italic>box¶

If the XPressTags contain style sheet calls, the style sheets are applied to the text.

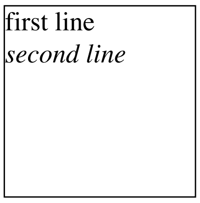
¶*T3*F2*$@Text:My <PB>text formatted as “Text”¶

creates a text box and attributes its content to the style sheet “Text”:


My **text format-
ted as “Text”**

¶*T3*F2*$@Text:first line
@Text italic:second line ¶

Creates a text box and formats the first text line with the style sheet “Text”, the second text line with the style sheet “text italic”:


first line
second line

If a style sheet is not present in the QuarkXPress document, it will be created.

¶*T3*F2*$@Text:My <@italic>box¶

Creates the style sheet “Text” and the character style sheet “italic” in a QuarkXPress document.

If the XPressTags also contain style sheet definitions, they will be applied on the newly created style sheet otherwise a standard style sheet is created. If a style sheet with the same name will be found in the QuarkXPress document, it will be applied, but not modified.

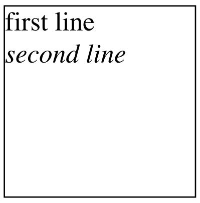
All texts that have been formatted with style sheets upon import can later be entirely reformatted by modifying the style sheet definitions.

Error search upon import

If you import the text with DATAformTags:

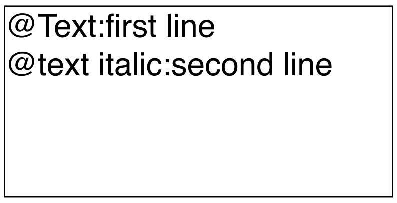
¶*T3*F2*$@Text:first line
@text italic:second line¶

the result is normally - and depending on style sheet definitions - the box:


first line
second line

The XPressTags are interpreted and translated into the style sheet assignments:

If the XPressTags filter is not used upon import, you get the text:


@Text:first line
@text italic:second line

The filter hasn’t been used; the XPressTags haven’t been interpreted, but read as a text.
In this case the Tag *{28}* was not **F2* or the XTension XPressTags Filter was missing. By the command Utilities/XTensions manager you may get an overview of the loaded XTensions and their status.

Unauthorised characters

Text with DATAformTags and style sheet names may not contain certain characters. Cf. points 5) and 6) of syntax rules in chapter II. DATAformTags:

5) unauthorised characters in a text with DATAformTags: < @ \
6) unauthorised characters in names: @ : " ¶ { } ; =

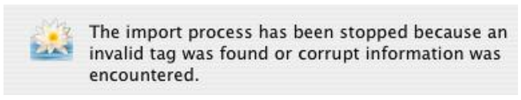
The characters < @ are reserved as markers for XPressTags. If necessary they must be masked by the relevant XPressTag.

The character "<" marks the beginning of an XPressTag. It must not be used to signify the smaller-than-sign. "@" marks the beginning of a style sheet call.
If these three characters must be present in the QuarkXPress text, they must be replaced in the database or on export by the relevant XPressTags:

<	by	<\<>
\	by	<\\>
@	by	<\@>

Example:
¶*T3*F2*$the distance is < 3mm.¶

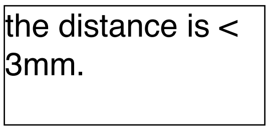
If this text with DATAformTags is transferred to QuarkXPress, the character < is interpreted by QuarkXPress as the start of an XPressTag. The entire text from the smaller-than-sign on, won't be displayed. QuarkXPress will show the error message:


The import process has been stopped because an invalid tag was found or corrupt information was encountered.

The correct text with DATAformTags should be:

¶*T3*F2*$the distance is <\<> 3mm.¶

This text creates the correct content:

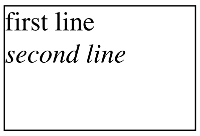

the distance is <
3mm.

Export

When exporting text boxes XPressTags are exported within the text if this has been specified in the DATAform preferences. (See under DATAformXTension "Preferences...", page: 25)
Depending on the settings in the preferences dialogue, the text is transferred with or without XPressTags. If the text is exported with XPressTags, all style sheets and style sheet definitions are also exported. In this case the exported text is more detailed than the imported text.
If you import e.g. in a first step the lines:

¶*T3*F2*$@Text:first line
<@italic>second line¶

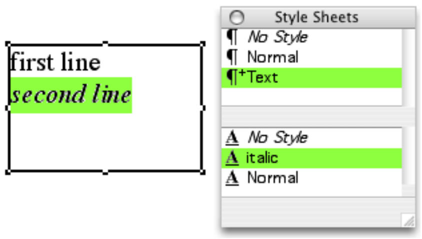
you will get the box:


first line
second line

If you now mark the box and export it with the DATAform command "Export selection", you will get a text file with the text with DATAformTags (If loading the text with the QuarkXPress command “Get Text...”, the option “Convert Quotes” must be switched off; see also under DATAformTags quick start, page: 11):

```
DfXT+2.0¶*T107*e["FEUER:em/DF:DF Kit:DATAformXTensionE";841.89;595.276;
28.346;28.346;28.346;28.346;0;1;22.677;0]*$¶*#0*G0*T3*@1*x0*y0*X72*Y47*W0*S0*
C1*c12.024*i0*I0*j0*J0*f0*H1*h1*D0*d0*B"White"*b0.1*F2*L128*l0.5*
M["Black";"White"]*m[1;1]*R0*r1*w0*s0*A0*a0*l0*»0*«0*§0*N0*-
0*g"222222"*$<v6.10><e0>
@Normal=<Ps100t0h100z10k0b0cKf"Helvetica">
@italic=<PIs100t0h100z10k0b0cKf"Times-Roman">
@Normal=[S"","Normal","Normal"]<*L*h"Standard"*kn0*kt0*ra0*rb0*d0*p(0,0,0,0,0,0,g,"
German")>
@Text=[S"","Text"]<*L*h"Standard"*kn0*kt0*ra0*rb0*d0*p(0,0,0,0,0,0,g,"German")*t(28.
346,0,"1  "56.693,0,"1  "85.039,0,"1  "113.386,0,"1  "170.079,0,"1
")Ps100t0h100z10k0b0cKf"Times-Roman">
@Text:first line
<@italic><a$$>second line¶
```

The italic text after the Tag *$ contains the style sheet definitions.
If the text is imported into a new QuarkXPress document with the DATAform command “Import boxes”, a text box is created, both style sheets are created defined according to the definitions and assigned to both lines. You get the following picture:

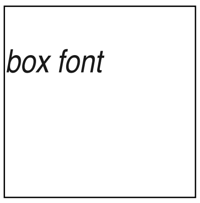


Box font and XPressTags

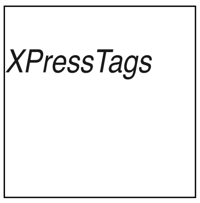
The table puts the nine Tags of the DATAform box font opposite the XPressTags for character attributes.

	DATAform Box font		XPressTags	
	Tag	Format	Format	Tag
Font	*__1__"Helvetica"	No. or Name	Name	<f"Helvetica">
standard	*__2__0	Number	XPressTag	<P>
bold	*__2__1	Number	XPressTag	<B>
italic	*__2__2	Number	XPressTag	<I>
underlined	*__2__4	Number	XPressTag	<U>
outline	*28	Number	XPressTag	<O>
shadow	*__2__16	Number	XPressTag	<S>
superscript	*__2__32	Number	XPressTag	<+>
subscript	*__2__64	Number	XPressTag	<=>
strike through.	*__2__512	Number	XPressTag	</>
all caps	*__2__1024	Number	XPressTag	<K>
small caps	*__2__2048	Number	XPressTag	<H>
word underline	*__2__4096	Number	XPressTag	<W>
Size	*__3__12	Point	Point	<z12>
Horizontal scale	*__4__0.85	from 1	Percent	<h85>
Colour	*__5__"Black"	No. or Name	Name	<c"Black">
Shade	*__6__0.9	from 1	Percent	<s90>
Kern amount	*__7__-20	1/200 Em spaces	1/200 Em spaces	<k-20>
Track amount	*__8__20	1/200 Em spaces	1/200 Em spaces	<t20>
Baseline shift	*__9__1.1	multiple of size	Point	<b13.2>

¶*T3*F1*1"Helvetica"*__2__2*__3__12*__4__0,85*__5__"Black"*__6__0,9*__7__-20*__8__20*__9__1.1*$box font¶
creates a text box and defines the text style by box font and *F1:


box font

¶*T3*F2*$<f"Helvetica"PIz12h85c"Black"s90k-20t20b13.2>XPressTags¶
creates the same text style by XPressTags and *F2:


XPressTags

Importing “hand made” documents

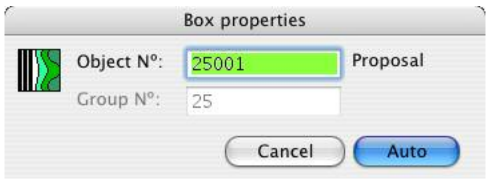
DATAform offers the possibility to save even complex QuarkXPress documents in the database. The individual modules (groups on boxes) are archived specifically and can be recreated in new arrangements in QuarkXPress 1:1. This type of archiving is not document or page related, but reaches the individual, context defined, module.

If boxes are placed in QuarkXPress manually or if an existing QuarkXPress document is to be transferred into the database, the single boxes must be given corresponding numbers. With these numbers the database will know the arrangement, the fellowship of the imported boxes. The boxes can then be saved accordingly. The numbers and box IDs inform the database: Which boxes, pictures and lines are part of the same article and constitue an article module? Which text box is the main box of the module, which others are additional text boxes?

Automatic box ID

The procedure of assigning numbers is simplified by DATAformXTension: All elements of an article module can be numbered with a single command.

To allow this you first group all a modules boxes to a QuarkXPress group. The boxes of this group are then numbered. You then activate the main text box of the group with the content tool and call the DATAform command “Box properties...”:

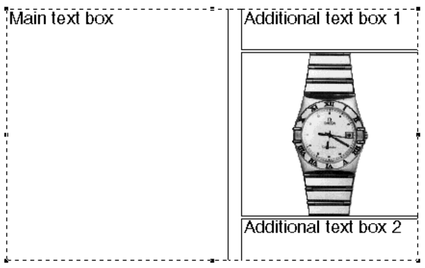

Box properties

Object N°: 25001 Proposal

Group N°: 25

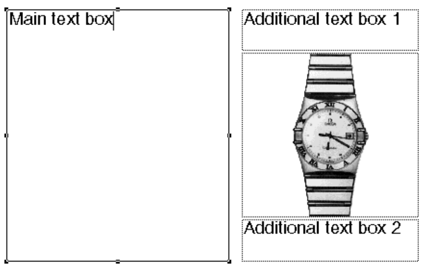
The Box properties dialogue suggests a number for this article module. It suggests the next highest free number in the QuarkXPress document.
The OK button is now called “Auto”; if you press “Auto” all boxes of the group will be numbered automatically. The main box is assigned the number 25001, the additional boxes are assigned the numbers 25101, 25102 and 25103 etc.

Example, step by step:
The following boxes were created manually and should be prepared for transfer into the database, i.e. they must be numbered:



The four boxes have been grouped i.e. these boxes should form a module. Select the QuarkXPress content tool:

and click beside the group first and then in the main box. The article module then shows the following status: Several boxes have been grouped and a text box has been activated with the content tool:



In this situation you open the Box properties dialogue and receive the Auto button as shown above.
(The Auto button only appears under these conditions.)

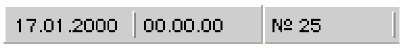
Importing "hand made" documents in 4 steps

To prepare an already existing, conventionally created document for transfer into the database you must first number all the article modules, export all the boxes and then import them into the database. The numbers of the article modules must be new and must not have been used previously in the database.

The number assignment and importing of conventionally created QuarkXPress documents into the DATAform database, taken here as an example of a DATAformXTension application, is performed with the following 4 main steps:

1. Search for next record number

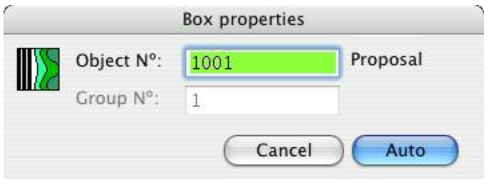
To receive the next free record number it is easiest to create a new record in your database.



In the footbar (in DATAform database) the desired number appears beside N°, in this case 25.

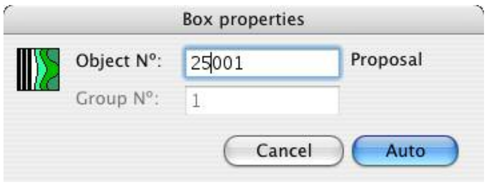
2. Prepare article modules, specify Box IDs

- Group all boxes of the first module as a QuarkXPress group.
- Deactivate the group by clicking beside it. Then activate the main text box with the content tool and call the menu command "Box properties...":

  Object N°: 1001 Proposal
  Group N°: 1

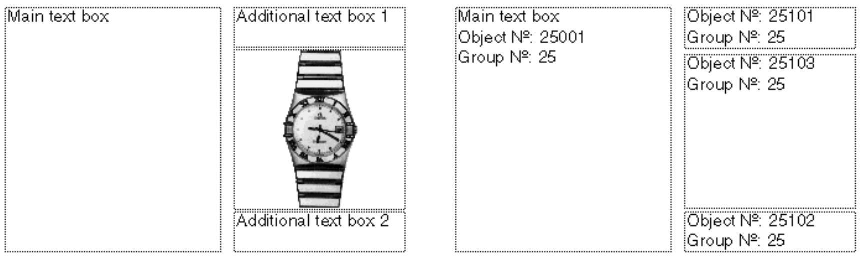
  The dialogue makes a proposal for the article number depending on the boxes in the Quark-XPress document. The proposal "1001" appears if the document does not yet contain any DATAform boxes.
- Now overwrite the module number with the first free number found in the database, in this example 25:
  The proposal for the object N° was "1001", the digits "001" are the ID for main text box, the module number is the number before "001". You therefore overwrite the "1" with "25" and get "25001":

  Object N°: 25001 Proposal
  Group N°: 1

The number of the entire module is now 25. This was also the number of the newly created record in the database. When importing the module into the database the module will be saved in this record Nº 25.
(The group number, in this case “1” was suggested, cannot be changed in Auto mode and will be adjusted automatically.)

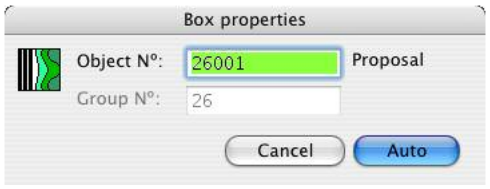
- Click on “Auto” and all boxes in the module will be numbered.



The illustration shows on the left an article module that contains four boxes and on the right the box IDs that were assigned to the individual boxes: 25001 for the main text box, 25101, 25102, 25103 etc. for all additional boxes.

3. Preparing further modules

Step 2 will then be performed on all other modules that you wish to transfer to the database. If the first module was assigned the Number “25”, the system then automatically suggests the number “26” for the next module:


Box properties

Object Nº: 26001 Proposal

Group Nº: 26

You click on “Auto” and all the modules boxes will be numbered.

Including XPressTags?

In the DATAform preferences dialogue in QuarkXPress you can specify whether the box texts should be exported with XPressTags or not. The option “Including XPressTags” exports all the font and style properties of the box texts and places them as XPressTags in the text. The use of this option has two possible drawbacks:
- The Tags may possibly hinder the readability of the texts.
- These styles are not easily modified by the database.

An export with XPressTags is however necessary if three conditions appear together:

a. The box text contains changing font styles.
b. This formatting cannot be generated as automatic formatting by the database.
c. All format properties of the text should be saved in the database.

If boxes are exported “Including XPressTags”, all style and font changes within the texts are retained.

4. Exporting the boxes and import into the database

Finally you export the prepared boxes from the QuarkXPress document and import them into DATAform with the command “Import elements...” with “Boxes” and “Content” set to ON.

Boxes that have not been numbered, i.e. their group number is zero, are skipped by the DATAform database; hence you can assign numbers to certain parts of the documents boxes and still use the command “Export all”.

The DATAform database is configured so that the elements that are not found in the database via their object number are created automatically as new records or elements in the element list.

Using the same method you can also attach new subboxes from QuarkXPress to already existing DATAform records.

Database design

The logic of box numbering is dependent on the database into which the articles will be imported. In the DATAform database the following procedure is used:

An article module consists, seen from a QuarkXPress point of view, from a main text box and any other boxes that together make up the article. From the database view point an article module is made up of a record with related records.
All DATAformTags, including the definitions for the main text box, are saved in the related records. The related records contain a field for the box ID whose value identifies the main record as well as the related records.

The main box of the module, i.e. the box with the article text, is assigned the unique number:
Record number * 1000 +1 in the example *#25001
All further boxes receive numbers from 25100 on i.e. *#25101, *#25102 etc.

With this method a module may theoretically contain 900 boxes without coming into conflict with another one. The database can therefore also attach individual boxes to the main record or update them; the main record can at any moment be identified as such.

For the transfer all boxes of a record receive the record number as the group number: Example *G25. They may then be combined as a group in QuarkXPress.

Order of objects

Order of QuarkXPress objects for DATAformXTension
The various functions act differently:

Export group/all

The order with which the boxes are exported by DATAformXTension is from the back to the front. The object farthest back is exported first, the foremost one last.
The order applies per page. Objects on more than one page are exported per page in ascending order.
The order is independent from whether the objects are grouped or not.

Both commands always use the order “from the back to the front”.

Export selection

The export order is usually the reverse of the order in which objects are clicked on; the last object clicked is exported first. However this is valid only if in the selection no objects are grouped and all objects are clicked individually with the shift key pressed. If the objects or some of the objects are selected by the selection tool, the export order is not clearly defined or is hard to predict.

This method is hardly suitable for the export of object layers.

Automatic assignment of box ID.

The assignment is performed from the back to the front as long as the group contains no further groups. The farthest additional box is assigned the number X101, then X102 up to the foremost box. The position of the main box has no influence on this.
If the group contains further groups, the order is not clearly defined or hard to predict.

This function keeps the order “from the back to the front” if no subgroups are present.

Adjust box height

If DATAformXTension adjusts the box heights of text boxes at the end of an import procedure, the text boxes will be processed in the order from the *front to the back*. This is necessary in order to take into account runaround properties of overlapping boxes.

Character set tables

Standard ASCII character set for all operating systems

9	Tabulator, HT
10	Linefeed, LF
13	Carriage Return, CR
32	Space bar

32		44	,	56	8	68	D	80	P	92	\	104	h	116	t
33	!	45	-	57	9	69	E	81	Q	93	]	105	i	117	u
34	"	46	.	58	:	70	F	82	R	94	^	106	j	118	v
35	#	47	/	59	;	71	G	83	S	95	_	107	k	119	w
36	$	48	0	60	<	72	H	84	T	96	`	108	l	120	x
37	%	49	1	61	=	73	I	85	U	97	a	109	m	121	y
38	&	50	2	62	>	74	J	86	V	98	b	110	n	122	z
39	'	51	3	63	?	75	K	87	W	99	c	111	o	123	{
40	(	52	4	64	@	76	L	88	X	100	d	112	p	124	|
41	)	53	5	65	A	77	M	89	Y	101	e	113	q	125	}
42	*	54	6	66	B	78	N	90	Z	102	f	114	r	126	~
43	+	55	7	67	C	79	O	91	[	103	g	115	s	127	□

Extended ASCII character set, MacOS / ANSI-Windows

128	Ä	196	160	†	134	192	¿	191	224	‡	135
129	Å	197	161	°	176	193	¡	161	225	·	183
130	Ç	199	162	¢	162	194	¬	172	226	‚	130
131	É	201	163	£	163	195	√	178	227	„	132
132	Ñ	209	164	§	167	196	ƒ	131	228	‰	137
133	Ö	214	165	•	149	197	≈	179	229	Â	194
134	Ü	220	166	¶	182	198	∆	185	230	Ê	202
135	á	225	167	ß	223	199	«	171	231	Á	193
136	à	224	168	®	174	200	»	187	232	Ë	203
137	â	226	169	©	169	201	…	46	233	È	200
138	ä	228	170	™	153	202		160	234	Í	205
139	ã	227	171	´	180	203	À	192	235	Î	206
140	å	229	172	¨	168	204	Ã	195	236	Ï	207
141	ç	231	173	≠	164	205	Õ	213	237	Ì	204
142	é	233	174	Æ	198	206	Œ	140	238	Ó	211
143	è	232	175	Ø	216	207	œ	156	239	Ô	212
144	ê	234	176	∞	129	208	–	150	240		173
145	ë	235	177	±	177	209	—	151	241	Ò	210
146	í	237	178	≤	141	210	“	147	242	Ú	218
147	ì	236	179	≥	142	211	”	148	243	Û	219
148	î	238	180	¥	165	212	‘	145	244	Ù	217
149	ï	239	181	µ	181	213	’	146	245	ı	208
150	ñ	241	182	∂	143	214	÷	247	246	ˆ	136
151	ó	243	183	∑	144	215	◊	190	247	˜	152
152	ò	242	184	∏	222	216	ÿ	255	248	¯	175
153	ô	244	185	π	254	217	Ÿ	159	249	˘	188
154	ö	246	186	∫	157	218	⁄	189	250	˙	221
155	õ	245	187	ª	170	219	€	128	251	˚	215
156	ú	250	188	º	186	220	‹	139	252	¸	184
157	ù	249	189	Ω	166	221	›	155	253	˝	158
158	û	251	190	æ	230	222	ﬁ	138	254	˛	240
159	ü	252	191	ø	248	223	ﬂ	154	255	ˇ	253

Extended ASCII character set, ANSI-Windows / MacOS

128	€	219	160		202	192	À	203	224	à	136
129	∞	176	161	¡	193	193	Á	231	225	á	135
130	‚	226	162	¢	162	194	Â	229	226	â	137
131	ƒ	196	163	£	163	195	Ã	204	227	ã	139
132	„	227	164	≠	173	196	Ä	128	228	ä	138
133	…	201	165	¥	180	197	Å	129	229	å	140
134	†	160	166	Ω	189	198	Æ	174	230	æ	190
135	‡	224	167	§	164	199	Ç	130	231	ç	141
136	ˆ	246	168	¨	172	200	È	233	232	è	143
137	‰	228	169	©	169	201	É	131	233	é	142
138	ﬁ	222	170	ª	187	202	Ê	230	234	ê	144
139	‹	220	171	«	199	203	Ë	232	235	ë	145
140	Œ	206	172	¬	194	204	Ì	237	236	ì	147
141	≤	178	173		240	205	Í	234	237	í	146
142	≥	179	174	®	168	206	Î	235	238	î	148
143	∂	182	175	¯	248	207	Ï	236	239	ï	149
144	∑	183	176	°	161	208	ı	245	240	˛	254
145	‘	212	177	±	177	209	Ñ	132	241	ñ	150
146	’	213	178	√	195	210	Ò	241	242	ò	152
147	“	210	179	≈	197	211	Ó	238	243	ó	151
148	”	211	180	´	171	212	Ô	239	244	ô	153
149	•	165	181	µ	181	213	Õ	205	245	õ	155
150	–	208	182	¶	166	214	Ö	133	246	ö	154
151	—	209	183	·	225	215	˚	251	247	÷	214
152	˜	247	184	¸	252	216	Ø	175	248	ø	191
153	™	170	185	∆	198	217	Ù	244	249	ù	157
154	ﬂ	223	186	º	188	218	Ú	242	250	ú	156
155	›	221	187	»	200	219	Û	243	251	û	158
156	œ	207	188	˘	249	220	Ü	134	252	ü	159
157	∫	186	189	⁄	218	221	˙	250	253	ˇ	255
158	˝	253	190	◊	215	222	∏	184	254	π	185
159	Ÿ	217	191	¿	192	223	ß	167	255	ÿ	216

Not all characters are displayed the same way with all fonts or are available there. The characters displayed above are from a Macintosh Times.

Tables of DATAformTags

DX_Array	Number of the Tag in the DATAform 4D interface
DATAform-Marke	Most characters look the same under MacOS and Windows. If not, e.g. with *{60} *Δ / Win *Æ,* the Windows representation (*Æ) is particulary mentioned.
=>	Tag is only imported in QuarkXPress
<=	Tag is only exported from QuarkXPress
•	Tag is also read by DX_Read (3) in the DX_Array, all others then in DX_Array {11}.
TEXT	The value can contain characters > ASCII 127: 1,3,25,29,33,40,63,69,73,74
ASCII	ASCII-value of the Tag, specified in the case of unclear characters.
Object delimiter	is ASCII 166, representation under MacOS = ¶, Windows = ¦ or ASCII 182, representation under MacOS = ∂, Windows = ¶

Sorted by {number}

This table is also contained in the DATAform 4D interface as DX_Parameter method.

` DX_Array			DATAformTag	Parameter	
`	{1}	*$	•	Text/Path	TEXT
`	{2}	*#	•	Object N°	
`	{3}	*P	• =>	Master page	"TEXT"
`	{4}	*G	•	Group N°	
`	{5}	*T	•	Object type	
`	{6}	*p	•	Page	
`	{7}	*x	•	Position left	
`	{8}	*X	•	Position right	
`	{9}	*y	•	Position top	
`	{10}	*Y	•	Position bottom	
`	{11}	*W		Box angle	
`	{12}	*S		Box skew	
`	{13}	*C		Columns	
`	{14}	*c		Gutter Width	
`	{15}	*i		Text inset left	
`	{16}	*I		Text inset right	ASCII 73
`	{17}	*j		Text inset top	
`	{18}	*J		Text inset bottom	
`	{19}	*N		Suppress printout	
`	{20}	*f		First baseline offset	
`	{21}	*H		Cap height	
`	{22}	*h		Accent	
`	{23}	*D		Vertical alignment type	
`	{24}	*d		Inter ¶ maximum	
`	{25}	*B		Box colour	"TEXT"
`	{26}	*b		Shade	
`	{27}	*-		Transparent	ASCII 45
`	{28}	*F	•	Font mode	
`	{29}	*1		Font	"TEXT"
`	{30}	*2		Font style	
`	{31}	*3		Font size	
`	{32}	*4		Font horizontal scale	
`	{33}	*5		Font/picture colour	"TEXT"
`	{34}	*6		Font/picture shade	

`	{35}	*7		Picture negative	
`	{35}	*7		Font kern amount	
`	{36}	*8		Font track amount	
`	{37}	*9		Font baseline shift	
`	{38}	*L		Frame/line style	
`	{39}	*l		Frame/line width	
`	{40}	*M[...]		Frame/line colour	"TEXT"
`	{41}	*m[...]		Frame/line shade	
`	{42}	*R		Runaround type	
`	{43}	*r		Runaround all	
`	{44}	*u		Runaround left	
`	{45}	*U		Runaround right	
`	{46}	*v		Runaround top	
`	{47}	*V		Runaround bottom	
`	{48}	*Q		Box shape	
`	{49}	*q		Corner radius	
`	{50}	*n		Suppress picture printout	
`	{51}	*Z	=>	Picture position	
`	{52}	*K		Picture scale across	
`	{53}	*k		Picture scale down	
`	{54}	*(		Picture offset across	
`	{55}	*)		Picture offset down	
`	{56}	*w		Picture angle	
`	{57}	*s		Picture skew	
`	{58}	*E		Arrowheads	
`	{59}	*§ / Win *¤		Locked	ASCII 164
`	{60}	*∆ / Win *Æ	=>	Adjust box height	ASCII 198
`	{61}	*x>	=>	Offset across	
`	{62}	*y>	=>	Offset down	
`	{63}	*+	• <=	Section page	"TEXT"
`	{64}	*» / Win *È	•	Chain: next box	ASCII 200
`	{65}	*« / Win *Ç	•	Chain: previous box	ASCII 199
`	{66}	*A	•	Anchor in box	
`	{67}	*a		Anchor: align with text	
`	{68}	*|		Flipping	ASCII 124
`	{69}	*e[…]		Blending	"TEXT"
`	{70}	*t[…]		Polygon	
`	{71}	*@	•	Character set table	
`	{72}	*±[...]		Trapping	ASCII 177
`	{73}	*?	•	Infotext	"TEXT"
`	{74}	*=[...]		Layer	"TEXT"
`	{75}	*g		Box properties	
`	{76}	*z		Picture clipping	
`	{77}	*,	=>	Additional Page	comma
`	{78}	*/[...]		Drop Shadow	
`	{79}	*~		Opacity	
`	{80}	*0		Picture Background	*zero

Sorted by *Tags

`	{2}	*#	•	Object Nº	
`	{1}	*$	•	Text/Path	TEXT
`	{54}	*(		Picture offset across	
`	{55}	*)		Picture offset down	
`	{63}	*+	• <=	Section page	"TEXT"
`	{77}	*,	=>	Additional Page	comma
`	{27}	*-		Transparent	ASCII 45
`	{78}	*/[...]		Drop Shadow	
`	{80}	*0		Picture Background	*zero
`	{29}	*1		Font	"TEXT"
`	{30}	*2		Font style	
`	{31}	*3		Font size	
`	{32}	*4		Font horizontal scale	
`	{33}	*5		Font/picture colour	"TEXT"
`	{34}	*6		Font/picture shade	
`	{35}	*7		Picture negative	
`	{35}	*7		Font kern amount	
`	{36}	*8		Font track amount	
`	{37}	*9		Font baseline shift	
`	{74}	*=[...]		Layer	"TEXT"
`	{73}	*?	•	Infotext	"TEXT"
`	{71}	*@	•	Character set table	
`	{66}	*A	•	Anchor in box	
`	{67}	*a		Anchor: align with text	
`	{25}	*B		Box colour	"TEXT"
`	{26}	*b		Shade	
`	{13}	*C		Columns	
`	{14}	*c		Gutter Width	
`	{23}	*D		Vertical alignment type	
`	{24}	*d		Inter ¶ maximum	
`	{58}	*E		Arrowheads	
`	{69}	*e[…]		Blending	"TEXT"
`	{28}	*F	•	Font mode	
`	{20}	*f		First baseline offset	
`	{4}	*G	•	Group Nº	
`	{75}	*g		Box properties	
`	{21}	*H		Cap height	
`	{22}	*h		Accent	
`	{16}	*I		Text inset right	ASCII 73
`	{15}	*i		Text inset left	
`	{18}	*J		Text inset bottom	
`	{17}	*j		Text inset top	
`	{52}	*K		Picture scale across	
`	{53}	*k		Picture scale down	
`	{38}	*L		Frame/line style	
`	{39}	*l		Frame/line width	
`	{40}	*M[...]		Frame/line colour	"TEXT"
`	{41}	*m		Frame/line shade	
`	{19}	*N		Suppress printout	
`	{50}	*n		Suppress picture printout	
`	{3}	*P	• =>	Master page	"TEXT"
`	{6}	*p	•	Page	
`	{48}	*Q		Box shape	

`	{49}	*q		Corner radius	
`	{42}	*R		Runaround type	
`	{43}	*r		Runaround all	
`	{12}	*S		Box skew	
`	{57}	*s		Picture skew	
`	{5}	*T	•	Object type	
`	{70}	*t[…]		Polygon	
`	{45}	*U		Runaround right	
`	{44}	*u		Runaround left	
`	{47}	*V		Runaround bottom	
`	{46}	*v		Runaround top	
`	{11}	*W		Box angle	
`	{56}	*w		Picture angle	
`	{8}	*X	•	Position right	
`	{7}	*x	•	Position left	
`	{61}	*x>	=>	Offset across	
`	{10}	*Y	•	Position bottom	
`	{9}	*y	•	Position top	
`	{62}	*y>	=>	Offset down	
`	{51}	*Z	=>	Picture position	
`	{76}	*z		Picture clipping	
`	{68}	*|		Flipping	ASCII 124
`	{79}	*~[...]		Opacity	
`	{59}	*§ / Win *¤		Locked	ASCII 164
`	{72}	*±[...]		Trapping	ASCII 177
`	{60}	*∆ / Win *Æ	=>	Adjust box height	ASCII 198
`	{65}	*« / Win *Ç	•	Chain: previous box	ASCII 199
`	{64}	*» / Win *È	•	Chain: next box	ASCII 200

Error numbers

Important system error messages

Not enough memory available. (-108)
Volume not found. (-53)
File is already open. (-49)
File name is already used. (-48)
File is busy (delete). (-47)
File is protected. (-45)
Volume is write protected. (-44)
File not found. (-43)
Too many files open. (-42)
End of file error. (-39)
File could not be opened. (-38)
File name is invalid. (-37)
File could not be read. (-36)
Volume not found. (-35)
Volume is full. (-34)

DATAform specific error messages

65537	A parameter error occurred.
65538	Not enough memory available.
65539	Box ID could not be created / read.
65540	DATAformTags-text is incorrect. (2)
65541	Page not found.
65542	File read/write error.
65543	Volume is full.
65544	A memory error occurred.
65545	DATAformTags-text is incorrect. (1)
65546	Missing DATAformTag.
65547	An internal error occurred. (1)
65548	An internal error occurred. (2)
65549	Box type has been changed.
65550	An internal error occurred. (3)
65551	An internal error occurred. (4)
65552	DATAform cannot import/export this box type.
65553	An internal error occurred. (6)
65554	An internal error occurred. (7)
65555	XPressTags-Filter was not found. (1)
65556	A communication error occurred. (1)
65557	A communication error occurred. (2)
65558	A communication error occurred. (3)
65559	A communication error occurred. (4)
65560	A communication error occurred. (5)
65561	XPressTags-Filter was not found/cannot be used. (2)
65562	Process has been aborted. (1)
65563	No box selected.
65564	The box type is missing.
65565	Box type is incorrect.
65566	A folder was not found.
65567	Tag identifier * expected.
65568	Quotation mark " is missing.
65569	A number has been expected.

65570	Chain to next box could not be build:
65571	Chain to previous box could not be build.
65572	Process has been aborted. (2)
65573	DATAformTags-text is missing.
65574	This text can not be interpreted.
65575	This DATAform XTension is outdated.
65576	End identifier ¶ is missing.
65577	Object Nº is already used:
65578	In the moment all DATAform licences are in use for this QuarkXPress licence:
65579	Serial number is incorrect.
65580	This DATAform XTension is outdated for this QuarkXPress document.
65581	The "TO"-message is invalid.
65582	The "FROM"-message is invalid.
65583	Message not found.
65584	There is no current QuarkXPress document.
65585	Message command is invalid.
65586	No DATAform objects with this group Nº found:
65587	No box selected.
65588	PARAMETER-item empty or missing.
65589	The box cannot be anchored. The target box/position could not be found:
65590	The Box is anchored, it cannot be replaced:

Index