WTQ_DEHILITE Applies only to highlighting introduced by the API. If highlighting exists in the client window before the writing tool session is initiated, it is handled according to operating environment conventions. Preexistent highlighting in the client window would normally be removed, since the highlighted text is being processed by the writing tool.

Structure

```
typedef struct
{
    WTMSGID msgid;
    WTCOUNTcount;
} WTQDEHILITE;
```

Types

WTQDEHILITE A structure containing the WTQ_DEHILITE message.

WTQDEHILITEP A pointer to the WTQDEHILITE structure.

Members

msgid WTQ_DEHILITE.

count The number of bytes (in API terms) to be dehighlighted. A count of zero indicates that all highlighting should be removed from the client window.

Data

None.

WTQ_GOTO

After a text block is returned, the API cursor is at the end of that text block. If the writing tool wishes to do anything with the text, it must first move the cursor to the correct position in the text query. The WTQ_GOTO command is used to move the cursor to another location in the text query.

Structure

```
typedef struct
{
    WTMSGID msgid;
    WTPOS       blockpos;
    WTCOUNTblocknum;
    WTPOS       offsetpos;
    WTCOUNToffsetloc;
} WTQGOTO;
```

Types

WTQGOTO	A structure containing the WTQ_GOTO message.
WTQGOTOP	A pointer to the WTQGOTO structure.

Members

msgid	WTQ_GOTO.
blockpos	The type of positioning for the *blocknum* parameter.
blocknum	The block number to go to.
offsetpos	The type of positioning for the *offsetloc* parameter.
offsetloc	The offset within the block.
Data	None.

WTQ_HILITE

Instructs the client application to highlight text at the current cursor position inside its window.

The cursor stays in its original position after the WTQ_HILITE command is performed by the client application. Only a WTQ_GOTO command or another WTQ_TEXTBLOCK and WTR_TEXTBLOCK pair can actually move the cursor.

When the client receives a WTQ_HILITE command, it should scroll the highlighted text so that it is visible in its window, if necessary. In most cases this text should appear in the top portion of the client window, leaving the bottom portion available for the display of writing tool information. However, since this may vary among applications and among platforms, writing tools should allow their location on the screen to be configured by the user.

Structure

```
typedef struct
{
    WTMSGID msgid;
    WTCOUNTcount;
} WTQHILITE;
```

Types

WTQHILITE	A structure containing the WTQ_HILITE message.
WTQHILITEP	A pointer to the WTQHILITE structure.

Members

msgid	WTQ_HILITE.
count	The number of bytes (in API terms) to highlight.

Data

None.

WTQ_INFOBLOCK

Normally, this information block needs to be sent only once at the beginning of each writing tool session. However, during user editing, the user may change some of the setup information that has been sent to the writing tool. If this occurs, the writing tool may issue another WTQ_INFOBLOCK message to request the updated information. The

writing tool sends the WTQ_INFOBLOCK message

Structure

```
typedef struct
{
    WTMSGID msgid;
    WTTOOL      tooltype;
    WTTEXT      suptext;
    WTTEXT      preftext;
    int             version;
    WTBOOL      resume;
    WTSIZE      size;
} WTQINFOBLOCK;
```

Types

WTQINFOBLOCK — A structure containing the WTQ_INFOBLOCK message.

WTQINFOBLOCKP — A pointer to the WTQINFOBLOCK structure.

Members

msgid — WTQ_INFOBLOCK.

tooltype — This parameter contains the general classification of the writing tool.

suptext — Each bit in this parameter corresponds to a text representation supported by the writing tool.

preftext — This parameter has only one bit set, indicating the preferred mode of text representation. This allows a writing tool to specify the text type it works with most easily.

version — This parameter contains the highest API version that is supported by the writing tool and is less than or equal to the version passed by the client in the WTC_INIT message. This is the revision level at which the API communication will take place.

resume — This parameter is set to WT_TRUE when the writing tool is resuming an infomation block query that is already in progress. It is set to WT_FALSE when a new query is being initiated.

size — This parameter contains the maximum size of the text block the client can

return in the WTR_INFOBLOCK reply.

Data

None.

WTQ_NEXTTEXTBLOCK

After an initial WTQ_TEXTBLOCK message has been sent, the writing tool may send additional WTQ_NEXTTEXTBLOCK messages to request additional text blocks until the initial query is completed. The writing tool will be notified that there is additional data to transmit by the value of the *endblock* parameter in the WTR_TEXTBLOCK message.

Structure

```
typedef struct
{
    WTMSGID msgid;
    WTSIZE     size;
} WTQNEXTTEXTBLOCK;
```

Types

WTQNEXTTEXTBLOCK

A structure containing the WTQ_NEXTTEXTBLOCK message.

WTQNEXTTEXTBLOCKP

A pointer to the WTQNEXTTEXTBLOCK structure.

Members

msgid — WTQ_NEXTTEXTBLOCK.

size — The maximum size block to be returned.

Data

None.

WTQ_READTEXTOBJECT

Writing tools use the WTQ_READTEXTOBJECT message to request text objects from the client. Text objects are references to larger portions of data not contained in the normal text stream. Not all file formats support text objects.

Structure

```
typedef struct
{
    WTMSGID msgid;
    WTCOUNTidnumber;
    WTBOOL       resume;
    WTSIZE     size;
} WTQREADTEXTOBJECT;
```

Types

WTQREADTEXTOBJECT

A structure containing the WTQ_READTEXTOBJECT message.

WTQREADTEXTOBJECTP

A pointer to the WTQREADTEXTOBJECT structure.

Members

msgid WTQ_READTEXTOBJECT.

idnumber The ID of the text object to read.

resume Set to WT_TRUE if the writing tool is resuming a previous query. Set to WT_FALSE if the writing tool is initiating a new query.

size The maximum size block the WTR_READTEXTOBJECT message should return. This value is in bytes.

Data

None.

WTQ_READTOOLDATA

Requests a writing tool data area from the client. If the entire data can't be returned in one reply, the writing tool may use this message, setting the *resume* parameter, to request the next block.

Structure

```
typedef struct
{
    WTMSGID     msgid;
    int         company;
    int         product;
    WTDAMODE    tdamode;
    WTBOOL      resume;
    WTSIZE      size;
} WTQREADTOOLDATA;
```

Types

WTQREADTOOLDATA

A structure containing the WTQ_READTOOLDATA message.

WTQREADTOOLDATAP

A pointer to the WTQREADTOOLDATA structure.

Members

msgid WTQ_READTOOLDATA.

company This parameter contains a unique ID for the company that owns the writing tool that stored the data. WordPerfect, Corel Corporation, will register all company codes for companies producing writing tools that are made generally available. These ID numbers will fall in the range below 0x8000. Numbers above this boundary may be used without prior registration. If a company's writing tools do not make use of the tool data areas, the company does not need to

register with WordPerfect.

product This parameter is reserved for use by the writing tool company to specify product and version ID numbers. This may be used to separate the data for multiple writing tools offered by the same company.

tdamode This parameter is a bit field. Only one bit in the field may be set, specifying the tool data area that is to be returned.

resume This parameter is set to WT_TRUE when the writing tool wishes to resume a currently incomplete tool data query.

size This parameter specifies the maximum block size that may be returned by the WTR_READTOOLDATA reply message.

Data

None.

WTQ_REPLACE

Whenever this command is sent, all highlighting introduced by the API is first removed from the client window. The size of the text block containing the cursor and any adjacent text blocks shrinks according to the number of bytes deleted. The block containing the cursor then increases by the size of the text inserted. The cursor remains at the position it was in before the WTQ_REPLACE message.

Structure

```
typedef struct
{
    WTMSGID msgid;
    WTLANG      language;
    WTCOUNTdepth;
    WTCOUNTcount;
    WTBOOL      newblock;
    WTSIZE      size;
} WTQREPLACE;
```

Types

WTQREPLACE A structure containing the WTQ_REPLACE message.

WTQREPLACEP A pointer to the WTQREPLACE structure.

Members

msgid WTQ_REPLACE.

language The language of the text block to replace.

depth The depth of this text replacement.

count The number of bytes (in API terms) to be deleted from the client's buffer. The bytes will be deleted from the API cursor position.

newblock Set to WT_TRUE if the replacement text should be contained in its own text block.

size The size of the text buffer to be inserted. The size of the text may not exceed the size given by the client in the *mysize* parameter of the WTR_INFOBLOCK message.

Data

The buffer to insert into the document.

WTQ_TEXTBLOCK

Writing tools use the WTQ_TEXTBLOCK message to request a block or region of text from the client. A text region is specified by its beginning and ending boundaries (from position A to position B in the document).

The fields in the WTQP_TEXTBLOCK structure allow the writing tool to specify in a very precise manner the text it wishes to receive from the client. This is done by specifying starting and ending boundaries for the text query. A boundary may be one of several types of text entities. The text boundary entity will be contained in the text query. If the starting boundary is the previous paragraph, the query will start with the first letter in that paragraph. Similarly, ending boundaries will be the end of the text entity. These text entities comprise a simple method of specifying a text boundary. They are only as accurate as the definition used by the client. The writing tool may extend a query if it finds the definition to be

inadequate for its needs. By combining the positioning field with a location field that is not zero, it is possible to move to any location within the document.

If the entire text request is contained in the reply message, the complete flag is set in the WTR_INFOBLOCK reply. Otherwise, the maximum number of bytes possible is returned. The writing tool must send another WTQ_TEXTBLOCK message, this time with the *qtype* parameter set to WTQRESUME. This process continues until the client is able to complete the original request, at which point it returns the final text block with the complete flag set.

Structure

```
typedef struct
{
    WTMSGID     msgid;
    WTUNIT          fromunit;
    WTPOS           frompos;
    WTCOUNT     fromloc;
    WTUNIT          tounit;
    WTPOS           topos;
    WTCOUNT     toloc;
    WTTBQTYPE qtype;
    WTBOOL          containers;
    WTBOOL          tracking;
    WTSIZE      size;
    Window          windowTool
} WTQTEXTBLOCK;
```

Types

WTQTEXTBLOCK — A structure containing the WTQ_TEXTBLOCK message.

WTQTEXTBLOCKP — A pointer to the WTQTEXTBLOCK structure.

Members

msgid — WTQ_TEXTBLOCK.

fromtext — The text unit to position from. The upper boundary of the text query will be the beginning of the text unit specified in the *from* parameters of this message. Specifying zero in this parameter is equivalent to specifying the client's default text unit.

frompos — The type of positioning.

fromloc — The distance to the desired position.

totext — The text unit to position to. Specifying zero in this parameter is equivalent to specifying the client's default text unit.

topos — The type of positioning.

toloc — The distance to the desired position.

qtype — The type of text query message sent.

containers — Set to WT_TRUE if sub-containers should be sent in the query.

tracking — Set to WT_TRUE if the client should attempt to track editing changes made directly by the user to its text buffer during the writing tool session.

size — The maximum block size (in bytes) the WTR_TEXTBLOCK reply should return.

windowTool — The window ID of the Writing Tool window. This parameter is specific to the UNIX platform.

Data

None.

WTQ_TLACTIVE

The writing tool sends the WTQ_TLACTIVE message whenever it regains activation.

Structure

```
typedef struct
{
    WTMSGID msgid;
} WTQTLACTIVE;
```

Types

WTQTLACTIVE — A structure containing the WTQ_TLACTIVE message.

WTQTLACTIVEP — A pointer to the WTQTLACTIVE structure.

Members

msgid — WTQ_TLACTIVE.

Data

None.

WTQ_UNDOREPLACE

The cursor position is first set to the location it was in before the WTQ_REPLACE command occurred. The effects of the WTQ_REPLACE command are then undone, leaving the cursor in its original position. This allows writing tools to undo a replacement without concern for prior formatting code positions. This is particularly necessary when the text is transmitted without the formatting information.

Structure

```
typedef struct
{
    WTMSGID msgid;
} WTQUNDOREPLACE;
```

Types

WTQUNDOREPLACE

A structure containing the WTQ_UNDOREPLACE message.

WTQUNDOREPLACEP

A pointer to the WTQUNDOREPLACE structure.

Members

msgid — WTQ_UNDOREPLACE.

Data

None.

WTQ_UNITINFO

Sent by a writing tool to a client, requesting the current status of the units of text in the client. This message needs to be sent only if the writing tool wishes to present a customized menu of text unit choices to the user.

Structure

```
typedef struct
{
    WTMSGID msgid;
} WTQUNITINFO;
```

Types

WTQUNITINFO — A structure containing the WTQ_UNITINFO message. message.

WTQUNITINFOP — A pointer to the WTQUNITINFO structure.

Members

msgid — WTQ_UNITINFO.

Data

None.

WTQ_WRITETEXTOBJECT

Writing tools use the WTQ_WRITETEXTOBJECT message to write text

objects in a document. Text objects are references to larger sections of data not contained in the normal text stream of a document. Writing tools may either insert new text objects, replace existing text objects, or append information to text objects in a document.

Structure

```
typedef struct
{
    WTMSGID msgid;
    WTCOUNTidnumber;
    WTBOOL       append;
    WTSIZE     size;
} WTQWRITETEXTOBJECT;
```

Types

WTQWRITETEXTOBJECT

A structure containing the WTQ_WRITETEXTOBJECT message.

WTQWRITETEXTOBJECTP

A pointer to the WTQWRITETEXTOBJECT structure.

Members

msgid — WTQ_WRITETEXTOBJECT.

idnumber — The ID of the text object to write.

append — Set to WT_TRUE if the client is to append the new information to existing information if it exists. Set to WT_FALSE if the client is to replace the existing information.

size

The size of the text buffer to be inserted. The size of the text may not exceed the size given by the client in the *mysize* parameter of the WTR_INFOBLOCK message.

Data

The text object data to be inserted into the document.

WTQ_WRITETOOLDATA

Writes writing-tool-specific data to the indicated client area. If an unsupported client area is indicated, no data will be written.

Structure

```
typedef struct
{
    WTMSGID     msgid;
    int         company;
    int         product;
    WTDAMODE    tdamode;
    WTBOOL      append;
    WTSIZE      size;
} WTQWRITETOOLDATA;
```

Types

WTQWRITETOOLDATA

A structure containing the WTQ_WRITETOOLDATA message.

WTQWRITETOOLDATAP

A pointer to the WTQWRITETOOLDATA structure.

Members

msgid — WTQ_WRITETOOLDATA.

company — The company ID of the block to be written.

product — The product and version information for the block.

tdamode — The writing tool area to which the data should be written.

append — Set to WT_TRUE if the information should be appended onto information already stored with the client. Otherwise, the existing information will be truncated.

size — The size (in bytes) of the data to be written. This size should not exceed what is specified in the *mysize* field returned by the WTR_INFOBLOCK reply.

Data

The data block to be written.

WTR_DEHILITE

The client responds to a WTQ_DEHILITE message with a WTR_DEHILITE message indicating the actual number of bytes (in API terms) that were dehighlighted.

Structure

```
typedef struct
{
    WTMSGID msgid;
    WTSTATUS    status;
    WTCOUNTcount;
} WTRDEHILITE;
```

Types

WTRDEHILITE — A structure containing the WTR_DEHILITE message.

WTRDEHILITEP — A pointer to the WTRDEHILITE structure.

Members

msgid — WTR_DEHILITE.

status — The return status from the client.

count — The number of bytes (in API terms) actually dehighlighted.

Data

None.

WTR_GOTO

The client sends the WTR_GOTO message in response to a WTQ_GOTO request by the writing tool. The new cursor location information is returned to the writing tool.

Structure

```
typedef struct
{
    WTMSGID msgid;
    WTSTATUS    status;
    WTCOUNTcount;
    WTSIZE      block;
    WTSIZE      offset;
} WTRGOTO;
```

Types

WTRGOTO	A structure containing the WTR_GOTO message.
WTRGOTOP	A pointer to the WTRGOTO structure.

Members

msgid	WTR_GOTO.
status	The return status from the client.
count	The number of bytes (in API terms) by which the client's position changed.
block	The block number of the new position.
offset	The offset of the new position.

Data

None.

WTR_HILITE

The client responds to a WTQ_HILITE message with a WTR_HILITE message indicating the actual number of bytes (in API terms) that were highlighted.

Structure

```
typedef struct
{
    WTMSGID msgid;
    WTSTATUS    status;
    WTCOUNTcount;
} WTRHILITE;
```

Types

WTRHILITE	A structure containing the WTR_HILITE message.
WTRHILITEP	A pointer to the WTRHILITE structure.

Members

msgid	WTR_HILITE.
status	The return status from the client.
count	The number of bytes (in API terms) that were actually highlighted.

Data

None.

WTR_INFOBLOCK

If the window text cannot be sent in the block size specified, the complete flag in the reply is set to WT_FALSE. If the tool wishes to get the complete text, it sends another WTQ_INFOBLOCK message, setting the resume flag to WT_TRUE. The parameters (other than the maximum block size) are ignored, and the next block of text is returned. This

continues until the complete window name is sent.

Structure

```
typedef struct
{
    WTMSGID     msgid;
    WTSTATUS        status;
    WTTEXT          textmode;
    WTDAMODE tdamode;
    WTLANG          userlang;
    WTLANG          textlang;
    WTBOOL          hastext;
    int             service;
    WTSIZE      mysize;
    WTBOOL          complete;
    WTSIZE      winsize;
} WTRINFOBLOCK;
```

Types

WTRINFOBLOCK — A structure containing the WTR_INFOBLOCK message.

WTRINFOBLOCKP — A pointer to the WTRINFOBLOCK structure.

Members

msgid — WTR_INFOBLOCK.

status — The return status from the client.

textmode — This parameter has one bit set, designating which text type will be used during the session. This must be one of the bits set in the *suptext* field of the WTQ_INFOBLOCK query and should be the type specified in the *preftext* field if it is supported.

tdamode — This parameter specifies which tool data area modes the client supports. Requests for tool data from an unsupported area will return as if no data were available. Attempts to write to an unsupported area will return a response that no bytes were written to the area.

userlang — The language in which the client is currently presenting its user interface.

textlang — The default language for the client's text.

hastext This parameter is set to WT_TRUE if the client has a default block of text it wishes to send to the writing tool.

service This parameter contains a number corresponding to the desired service within the writing tool.

mysize This parameter specifies the maximum size block the client can receive from the writing tool. This is used by the WTQ_WRITETOOLDATA command when a tool data area is written to the client and in the WTQ_REPLACE command when replacement text is transferred.

complete This parameter is set to WT_TRUE if the reply completes the transmission of the client's document or window name.

winsize This parameter contains the size of the window name text.

Data

The name of the document or window.

WTR_READTEXTOBJECT

The requested text object is returned. If the size of the text object is greater than the maximum block size specified in the WTQ_READTEXTOBJECT message, the complete flag is set to WT_FALSE to inform the writing tool that there is additional information.

Structure

```
typedef struct
{
    WTMSGID msgid;
    WTSTATUS    status;
    WTBOOL      complete;
    WTSIZE      size;
} WTRREADTEXTOBJECT;
```

Types

WTRREADTEXTOBJECT

A structure containing the WTR_READTEXTOBJECT message.

WTRREADTEXTOBJECTP

A pointer to the WTRREADTEXTOBJECT structure.

Members

msgid — WTR_READTEXTOBJECT.

status — The return status from the client.

complete — Set to WT_TRUE if the entire text object has been sent. Set to WT_FALSE if there is additional data to be sent.

size — The actual size of the block sent. This may not exactly correspond with the requested size, because the client should not send a partial code at the end of a block. Clients may optionally split blocks on word or other boundaries, but this is not required by the API.

Data

The text object requested.

WTR_READTOOLDATA

If the entire data block does not fit in the maximum block size given, the writing tool sends another WTQ_READTOOLDATA message, setting the resume flag to indicate that the current query should be resumed. Subsequent blocks will be sent with the WTR_READTOOLDATA message, and the complete flag will be set when all information has been transferred. The product and document tool data areas may be requested at any time, but the cursor must be adjacent to a cursor data area to request it.

Structure

```
typedef struct
{
    WTMSGID msgid;
    WTSTATUS    status;
    WTBOOL      complete;
    WTSIZE      size;
} WTRREADTOOLDATA;
```

Types

WTRREADTOOLDATA — A structure containing the WTR_READTOOLDATA message.

WTRREADTOOLDATAP

A pointer to the WTRREADTOOLDATA structure.

Members

msgid — WTR_READTOOLDATA.

status — The return status from the client.

complete — Set to WT_TRUE if the data sent with the reply completes the transmission of the tool data area requested.

size — The actual size of the return data block.

Data

The data block requested.

WTR_REPLACE

The client sends the WTR_REPLACE message in response to a WTQ_REPLACE message.

Structure

```
typedef struct
{
    WTMSGID msgid;
    WTSTATUS    status;
    WTCOUNTcount;
    WTSIZE      size;
    WTSIZE      block;
    WTSIZE      offset;
} WTRREPLACE;
```

Types

WTRREPLACE	A structure containing the WTR_REPLACE message.
WTRREPLACEP	A pointer to the WTRREPLACE structure.

Members

msgid	WTR_REPLACE.
status	The return status from the client.
count	The actual number of bytes (in API terms) that were deleted.
size	The actual size of the buffer inserted.
block	The new block number.
offset	The new offset.

Data

None.

WTR_TEXTBLOCK

Each block in the text query is assigned a number in the order it is transferred, beginning with block zero. This message is sent in response to either the WTQ_TEXTBLOCK or WTQ_NEXTTEXTBLOCK messages.

Structure

```
typedef struct
{
    WTMSGID     msgid;
    WTSTATUS        status;
    WTSIZE          blocknum;
    WTLANG          language;
    WTCOUNT     depth;
    WTENDBLOCK  endblock;
    WTSIZE          size;
} WTRTEXTBLOCK;
```

Types

WTRTEXTBLOCK — A structure containing the WTR_TEXTBLOCK message.

WTRTEXTBLOCKP — A pointer to the WTRTEXTBLOCKstructure.

Members

msgid — WTR_TEXTBLOCK.

status — The return status from the client.

blocknum — The ID number of the text block returned in this message.

language — The language of the current text block. As the text language of the document may change at any time, this should be checked for a change whenever a text block is returned.

depth — If sub-containers were requested, this indicates the depth of the current text in relation to the previous text block.

endblock — The reason the block of text ended.

size — The actual size of the text block. This may not exactly correspond to the requested size, because the client should not send a partial code at the end of a block. Clients may optionally split blocks on word or other boundaries, but this is not required by the API.

Data

The text block requested.

WTR_TLACTIVE

Informs the writing tool of any changes that were made while the client was active.

Structure

```
typedef struct
{
```

```
    WTMSGID msgid;
    WTSTATUS    status;
    WTCOUNTmodified;
    WTCOUNTcursor;
    WTINFO      newinfo;
} WTRTLACTIVE;
```

Types

WTRTLACTIVE — A structure containing the WTR_TLACTIVE message.

WTRTLACTIVEP — A pointer to the WTRTLACTIVE structure.

Members

msgid — WTR_TLACTIVE.

status — The return status from the client.

modified — The number of text blocks (starting with the most recent) that were modified by the user while the client was active. If no editing changes were made, a zero is sent.

cursor — The number (starting with the most recent) of text blocks away from the current block the user moved the cursor. If the cursor is in the current block, a zero is sent.

newinfo — Set to WTTRUE when the user has made changes in the client that require the retransmission of the infomation block. This occurs, for example, when the user saves the current document under a new name or changes the interface language within the client. At this point the writing tool may either use the WTQ_INFOBLOCK message to request a new block of information or continue the text query with the outdated information.

Data

None.

WTR_UNITINFO

Returns information about the units of text available in the client application. This allows writing tools to customize the choices of text they offer to the user.

Structure

```
typedef struct
{
    WTMSGID msgid;
    WTSTATUS    status;
    WTUNIT      supunit;
    WTUNIT      curunit;
    WTUNIT      defunit;
    WTSIZE      size;
} WTRUNITINFO;
```

Types

WTRUNITINFO	A structure containing the WTR_UNITINFO message.
WTRUNITINFOP	A pointer to the WTRUNITINFO structure.

Members

msgid	WTR_UNITINFO.
status	The return status from the client.
supunit	Each bit in this parameter is set if the corresponding text unit is supported by the client application.
curunit	Each bit in this parameter is set if the corresponding text unit is currently available in the client application.
defunit	This parameter has only one bit set, indicating which text unit is currently considered the default text unit by the client application.
size	The size of the return buffer containing the client-defined units.

Data

The client-defined units. The buffer contains one null-terminated string in WTX_NATIVE format for each bit set in the upper half of the *supunit* parameter. The first string corresponds to bit 16, the second to bit 17, and so on.

WTR_WRITETEXTOBJECT

The client sends the WTR_WRITETEXTOBJECT message in response to a WTQ_WRITETEXTOBJECT message. The client indicates the status of the request and the size of the buffer inserted into the document. If the file format of the client does not support text objects, the size returned will be zero.

Structure

```
typedef struct
{
    WTMSGID msgid;
    WTSTATUS    status;
    WTSIZE     size;
} WTRWRITETEXTOBJECT;
```

Types

WTRWRITETEXTOBJECT

A structure containing the WTR_WRITETEXTOBJECT message.

WTRWRITETEXTOBJECTP

A pointer to the WTRWRITETEXTOBJECT structure.

Members

msgid — WTR_WRITETEXTOBJECT.

status — The return status from the client.

size — The actual size of the buffer inserted into the client document.

Data

None.

WTR_WRITETOOLDATA

The writing tool client responds to the WTQ_WRITETOOLDATA message by sending this message. The status of the message is returned as well as the actual size of the data that was written.

Structure

```
typedef struct
{
    WTMSGID msgid;
    WTSTATUS    status;
    WTSIZE      size;
} WTRWRITETOOLDATA;
```

Types

WTRWRITETOOLDATA

A structure containing the WTR_WRITETOOLDATA message.

WTRWRITETOOLDATAP

A pointer to the WTRWRITETOOLDATA structure.

Members

msgid — WTR_WRITETOOLDATA.

status — The return status from the client.

size — The actual number of bytes written.

Data

None.