# About the Corel Paradox Internet tools

The Corel Paradox Internet tools include several components that help end-users and developers publish HTML documents and communicate with Internet Web browsers:

- Hyperlink capabilities
- the HTML publishing features in Corel Paradox 8
- the Web Server Repository, a set of data tables that store templatized input for dynamic output in HTML form.
- the Corel Web Server (.EXE)
- the Corel Web Server Control ActiveX control
- the GXPublish DLL, an OLE Automation server that handles HTTP requests. GXPublish works with the Web Server Repository and compatible Corel Web Servers to update and present dynamic HTML documents.
- the GXEngine DLL, an OLE Automation server that outputs templatized Borland Database Engine (BDE) data in HTML format. It can be controlled with OLE Automation methods and properties and a custom ObjectPAL Library, the HTML Publishing Library.

  The following components help developers access the GXEngine:

  - the GXEngine Template Specification
  - the GXEngine Interface Specification (for static HTML documents)
  - the HTMLIB01 Corel Paradox HTML Publishing Library (for static and dynamic HTML documents)

As an example of how the Corel Paradox Internet tools work together, consider this scenario:

A Corel Paradox user, Art Lee, publishes a report, CUSTOMER.RSL. He uses the Corel Paradox HTML Report Expert to publish it as a dynamic document at the URL: `"Customers.HTM"`.

When the report is published, Corel Paradox builds a template representation of the report document and stores all necessary information about the document in the Web Server Repository.

A sales representative in Denmark requests this document from the Corel Web Server Control running in a Corel Paradox form. She makes this HTTP request by typing the following URL into her Web browser:
`http://MyMachine.Corel.com/Customers.HTM`

The Corel Web Server Control builds the OLE Gateway Interface (OGI) interface to the request and sends it to its internal Request Manager. The Request Manager looks in the Web Server Repository to see if the URL is mapped to any type of handler. In this case, it is mapped to GXPublish, so the HTTP request is sent off to GXPublish. GXPublish looks in the Web Server Repository for the information it needs to dynamically generate the resultant content for the request. It does so and sends the result back through the Corel Web Server Control to the browser.

---

**{button ,AL(`INTRO;COMPONENT;',0,"Defaultoverview",)} Related Topics**

# About the GXEngine DLL

This OLE Automation server handles the task of outputting Borland Database Engine (BDE) data in HTML format. It can be called directly from ObjectPAL or any other language that supports calling OLE Automation servers. It parses HTML documents for BDE-specific template tags, and replaces them with snapshots of the data sources. These templates are generated by Corel Paradox and stored in the Web Server Repository. Custom templates can be constructed using any text editor.

The GXEngine supports

- production of HTML pages with embedded, dynamically-updated BDE data
- full grouping and sort control
- full data display formatting control
- linked cursors with arbitrary nesting levels
- support for data-defined Tables of Contents and hyperlinks
- support for drill-down reporting
- HTML layout-driven data-fetching metaphor

See the following topics for more information about BDE HTML templates and their conversion to static HTML documents using the GXEngine Interface:

- the GXEngine Template Specification
- the GXEngine Interface Specification

To produce dynamic HTML documents, use the HTMLIB01 Corel Paradox HTML Publishing Library.

---

**{button ,AL(`OVERVIEW;GXENGINE;',0,"Defaultoverview",)}** **Related Topics**

# GXEngine Template Specification

A GXEngine template is a set of HTML tags that can be placed in any new or existing HTML document. When the document is passed through the GXEngine, these tags are replaced with appropriate HTML code and raw data from BDE (Borland Database Engine) tables. Any tag with invalid syntax is ignored by the GXEngine parser, as it would be by a Web browser.

A GXEngine template and BDE tables (described in the GXEngine Interface Specification) are added to the GXEngine with the AddTemplateFile and AddTemplateString methods. These methods are called before the GXEngine's Execute method is called.

---

**{button ,AL(`GXENGINE;GXTEMPSPEC;',0,"Defaultoverview",)} Related Topics**

# GXEngine template syntax

The syntax used in defining the GXEngine Template Specification is modeled after the SQL Statement and Function Reference in the Borland InterBase Workgroup Server Language Reference.

### Typefaces and special characters

| | |
|---|---|
| UPPERCASE | Keywords that must be typed exactly as they appear when used |
| *italic* | Parameters that may or may not be broken into smaller units (as per following chart) |
| [ ] | Square brackets enclose optional syntax. |
| ... | Closely spaced ellipses indicate that a clause within brackets can be repeated as many times as necessary. |
| \| | The pipe symbol indicates that either of the two syntax clauses that it separates may be used, but not both. Inside curly braces, the pipe symbol separates multiple choices, one of which must be used. If more than one of these syntaxes appear, only one is used. Precedence is determined left to right per the tag syntax definition (parameter order is ignored). |
| {} | Curly braces indicate that one of the enclosed options must be included in actual statement use. |

### HTML template tag arguments

The following values appear in *italics* and are used as arguments for the various HTML template tags:

| | |
|---|---|
| *html* | (any valid HTML text) |
| *string* | (all printable characters enclosed in quotation marks) |
| *fileref* | *string* (any valid HTML file name) |
| *sourceref* | *string* (reference name for data source) |
| *fieldref* | *string* (field name from *sourceref*; "#" refers to sequence number) |
| *fieldset* | "*fieldref* [, *fieldref* ... ]" (leading/trailing spaces ignored) |
| *templateref* | *string* (reference name for stored template) |
| *paramref* | *string* (reference name for stored parameter) |
| *indexref* | *string* (reference name for an index construct) |
| *indexfield* | *string* {"LOW" \| "HIGH" \| "HREF" \| "RANGE"} |
| *tabletags* | (any valid HTML table tags) |
| *dateval* | {DAY \| WEEK \| MONTH \| QUARTER \| YEAR} |
| *sortref* | {ASC \| DESC \| NONE} |
| *breakref* | {SINGLE \| MULTI \| ATBREAK} |
| *onoff* | {ON \| OFF} |
| *formatspec* | (Standard Delphi format declaration) |

---

**{button ,AL(`GXTEMPSPEC;',0,"Defaultoverview",)}** **Related Topics**

# GXEngine template tags

The following HTML template tags are defined in the GXEngine Template Specification:

| | |
|---|---|
| BDE_TABLE | Outputs an HTML table containing all of the data in the current SRC data source, including a header and column for each field specified in FLDS |
| BDE_SCAN | Marks a section to be repeated for each record in the current SRC data source |
| BDE_FIELD | Outputs data from SRC field or stored parameter |
| BDE_GROUP | Marks a section to be repeated for each block of records that meets the group criteria |
| BDE_INDEX | Marks a section as an Index (Table of Contents) construct |
| BDE_BREAK | Marks a segment break for an Index (Table of Contents) construct |
| BDE_INCLUDE | Appends another template at the current position |

---

**{button ,AL(`GXTEMPSPEC;',0,"Defaultoverview",)} Related Topics**

# BDE_TABLE (HTML template tag)

Outputs an HTML table containing all of the data in the current SRC data source, including a header and column for each field specified in FLDS.

### Syntax

```
<BDE_TABLE SRC=sourceref [FLDS=fieldset] [ENCODE=onoff] [INDEX=fieldset] [tabletags ... ]>
```

### Parameters

| | |
|---|---|
| SRC | Data source to scan |
| FLDS | Field(s) to output (default: all fields) |
| ENCODE | Encode the field output changing all illegal characters to URL escape sequences (for example, change "#" into "%42") |
| INDEX | Fields to be used in the generated INDEX (if nested in BDE_INDEX template tag) |
| *tabletags* | Any additional tags to be placed in the HTML table declaration <TABLE *tabletags*> |

### User examples

```
<BDE_TABLE SRC="ORDERS">
<BDE_TABLE SRC="ORDERS" FLDS="OrderNo,Date">
```

### Template input

```
<BDE_TABLE SRC="CUSTOMER" FLDS=”Name,Phone” BORDER=2>
```

### HTML output

```
<TABLE BORDER=2>
<TR><TH>Name</TH><TH>Phone</TH></TR>
<TR><TD>Adam Anderson</TD><TD>867-5309</TD></TR>
<TR><TD>Billy Watson</TD><TD>431-1000</TD></TR>
```

... repeat for each record ...

```
<TR><TD>William Gladstone</TD><TD>1-800-628-5560</TD></TR>
<TR><TD>Zorba Quincy</TD><TD>879-2000</TD></TR>
</TABLE>
```

### Web browser output

| Name | Phone |
|---|---|
| Adam Anderson | 867-5309 |
| Billy Watson | 431-1000 |

... repeat for each record ...

| | |
|---|---|
| William Gladstone | 1-800-628-5560 |
| Zorba Quincy | 879-2000 |

---

**{button ,AL(`GXTEMPSPEC;GXTOKEN;',0,"Defaultoverview",)}** **Related Topics**

# BDE_SCAN (HTML template tag)

Marks a section to be repeated for each record in the current SRC data source.

### Syntax

```
<BDE_SCAN SRC=sourceref FLDS=fieldset SORT="+,-,...">html</BDE_SCAN>
```

### Parameters

| | |
|---|---|
| SRC | Data source to scan |
| FLDS | Fieldset to use for grouping criteria (same as BDE_GROUP) |
| SORT | Sort order for each field in FLDS; + = ascending and – = descending |

### User examples

```
<BDE_SCAN SRC=”ORDERS”>
Hello there! This will print once for each record!<BR>
</BDE_SCAN>
```

### Template input

```
<BDE_SCAN SRC=”CUSTOMER”>
<BDE_TABLE SRC=”ORDERS” FLDS=”CustNo,OrderNo,Total” BORDER=2>
</BDE_SCAN>
```

### HTML output

```
<TABLE BORDER=2>
<TR><TH>CustNo</TH><TH>OrderNo</TH><TH>Total</TH></TR>
<TR><TD>1000</TD><TD>1021</TD><TD>$300.04</TD></TR>
<TR><TD>1000</TD><TD>2300</TD><TD>$320.45</TD></TR>
<TR><TD>1000</TD><TD>2304</TD><TD>$45.86</TD></TR>
</TABLE>
<TABLE BORDER=2>
<TR><TH>CustNo</TH><TH>OrderNo</TH><TH>Total</TH></TR>
<TR><TD>1001</TD><TD>1180</TD><TD>$22.88</TD></TR>
<TR><TD>1001</TD><TD>1182</TD><TD>$900.08</TD></TR>
<TR><TD>1001</TD><TD>1345</TD><TD>$623.55</TD></TR>
<TR><TD>1001</TD><TD>1500</TD><TD>$1045.99</TD></TR>
</TABLE>
```

... one orders table frame for each customer filtered on CustNo ...

```
<TABLE BORDER=2>
<TR><TH>CustNo</TH><TH>OrderNo</TH><TH>Total</TH></TR>
<TR><TD>5036</TD><TD>1280</TD><TD>$556.81</TD></TR>
<TR><TD>5036</TD><TD>1500</TD><TD>$44.92</TD></TR>
</TABLE>
```

### Web browser output

| CustNo | OrderNo | Total |
|---|---|---|
| 1000 | 1021 | $300.04 |
| 1000 | 2300 | $320.45 |
| 1000 | 2304 | $45.86 |

| CustNo | OrderNo | Total |
|---|---|---|
| 1001 | 1180 | $22.88 |
| 1001 | 1182 | $900.08 |
| 1001 | 1345 | $623.55 |
| 1001 | 1500 | $1045.99 |

... one orders tableframe for each customer ...

| CustNo | OrderNo | Total |
|---|---|---|
| 5036 | 1280 | $556.81 |
| 5036 | 1500 | $44.92 |

{button ,AL(`GXTEMPSPEC;GXTOKEN;',0,"Defaultoverview",)} Related Topics

# BDE_FIELD (HTML template tag)

Outputs data from SRC field or stored parameter.

### Syntax

```
<BDE_FIELD {[SRC=sourceref] [FLDS=fieldset] [INDEX=fieldref] [FORMAT=formatspec] |
   PARAM=fieldref} [ENCODE=onoff]>
```

### Parameters

| | |
|---|---|
| SRC | Data source (default: current innermost SRC) |
| FLDS | Field(s) to output |
| | (SRC=*tableref* - default: Blank) |
| | (SRC=*grouppref* - default: "LOW") |
| INDEX | Values to be used in the generated INDEX (if in BDE_INDEX template tag) |
| FORMAT | Standard Delphi format string (applies to Date, Time, DateTime, and Numeric type fields) |
| PARAM | Output stored parameter value |
| ENCODE | Encode the field output changing all illegal characters to URL escape sequences (for example, change "#" into "%42") |

### User examples

```
<BDE_GROUP NAME=”Each10” SRC=”CUSTOMER” FLDS=”CustNo” INCR=10>
<BDE_FIELD PARAM=”WELCOMETEXT”>
<BDE_SCAN SRC=”CUSTOMER”>
Customers from <BDE_FIELD SRC=”Each10” FLDS="LOW"> to <BDE_FIELD SRC=”Each10” FLDS="HIGH"><P>
Customer Number: <BDE_FIELD SRC=”CUSTOMER” FLDS=”CustNo”>
<BDE_SCAN SRC=”ORDERS”>
Order Number: <BDE_FIELD FLDS=”OrderNo”>
</BDE_SCAN></BDE_SCAN></BDE_GROUP>
```

### Template input

```
<BDE_SCAN SRC=”CUSTOMER”>
<B>Customer Number: <BDE_FIELD FLDS=”CustNo”></B><BR>
<BDE_SCAN SRC=”ORDERS”>
Order Number: <BDE_FIELD FLDS=”OrderNo”><BR>
</BDE_SCAN>
</BDE_SCAN>
```

### HTML output

```
<B>Customer Number: 1000</B><BR>
Order Number: 1021<BR>
Order Number: 2300<BR>
Order Number: 2304<BR>
<B>Customer Number: 1001</B><BR>
Order Number: 1180<BR>
Order Number: 1182<BR>
Order Number: 1345<BR>
Order Number: 1500<BR>
```

... one order list for each customer filtered on CustNo ...

```
<B>Customer Number: 5036</B><BR>
Order Number: 1280<BR>
Order Number: 1500<BR>
```

### Web browser output

**Customer Number: 1000**
Order Number: 1021
Order Number: 2300
Order Number: 2304
**Customer Number: 1001**
Order Number: 1180
Order Number: 1182
Order Number: 1345
Order Number: 1500

... one order list for each customer filtered on CustNo ...

**Customer Number: 5036**
Order Number: 1280
Order Number: 1500

---

**{button ,AL(`GXTEMPSPEC;GXTOKEN;',0,"Defaultoverview",)}** **Related Topics**

# BDE_GROUP (HTML template tag)

Marks a section to be repeated for each block of records that meets the group criteria.

### Syntax

```
<BDE_GROUP [NAME=groupref] [SRC=sourceref] FLDS=fieldset [{CHARS=integer [INCR=integer] |
   DATE=dateval | INCR=integer }] [SORT=sortref]> html </BDE_GROUP>
```

### Parameters

| | |
|---|---|
| NAME | Name to give the temporary data source this group creates |
| SRC | Data source to group (default: current innermost SRC) |
| FLDS | Field to use for grouping criteria (only the first field in the list is used for grouping; any additional fields are used to sort results) |
| CHARS | Number of characters to group by, starting with the first character |
| DATE | Type of date grouping (DAY, WEEK, MONTH, QUARTER, YEAR) |
| INCR | Range to consider in each group; not used when grouping on date fields |
| | INCR used with CHARS indicates ordinal character value to break, (for example, FLDS=”Name” CHARS=1 INCR=5 outputs group:A-E, group:F-J, etc...) |
| SORT | Sorting method to use for the first field; additional fields listed in FLDS will be sorted ascending (default: NONE) |

### User examples

```
<BDE_GROUP SRC=”CUSTOMER” FLDS=”LastName,FirstName” CHARS=20>
<BDE_GROUP SRC=”CUSTOMER” FLDS=”CustNo” INC=10>
<BDE_GROUP NAME=”DANGER” FLDS=”CustNo” INC=1>
```

### Template input

```
<BDE_GROUP NAME=”ByFirstChar” SRC=”CUSTOMER” FLDS=”FirstName” CHARS=1>
<B>Customers from <BDE_FIELD SRC=”ByFirstChar” FLDS="LOW"> to <BDE_FIELD SRC="ByFirstChar"
FLDS="HIGH"></B><BR>
<BDE_SCAN SRC=”CUSTOMER”>
Customer Name: <BDE_FIELD SRC=”CUSTOMER” FLDS=”Name”><BR>
</BDE_SCAN><P>
</BDE_GROUP>
```

### HTML output

```
<B>Customers from Adam to Anders</B><BR>
Customer Name: Adam Waldsworth<BR>
Customer Name: Alan Baldwin<BR>
Customer Name: Alex Rosenburg<BR>
Customer Name: Anders Hepsibah<BR><P>
<B>Customers from Bill to Boris</B><BR>
Customer Name: Bill Brigdon<BR>
Customer Name: Bill Fudsworth<BR>
Customer Name: Boris Hodgekins<BR>
```

... repeat for each name in each group ...

### Web browser output

**Customers from Adam to Anders**
Customer Name: Adam Waldsworth
Customer Name: Alan Baldwin
Customer Name: Alex Rosenburg
Customer Name: Anders Hepsibah
**Customers from Bill to Boris**
Customer Name: Bill Brigdon
Customer Name: Bill Fudsworth
Customer Name: Boris Hodgekins

... repeat for each name in each group ...

---

**{button ,AL(`GXTEMPSPEC;GXTOKEN;',0,"Defaultoverview",)} Related Topics**

# BDE_INDEX (HTML template tag)

Marks a section as an Index (Table of Contents) construct.

### Syntax

```
<BDE_INDEX [NAME=indexref] [TEMPLATE=templateref] [FILES=breakref] [HEAD=templateref]
   [FOOT=templateref]>html</BDE_INDEX>
```

### Parameters

| | |
|---|---|
| NAME | Name to give the temporary data source this index creates |
| TEMPLATE | Template name to construct INDEX (default: HTML Engine IndexTemplate property) |
| FILES | SINGLE: Index contents appended to single file |
| | MULTI: Index contents appended to single file; new file generated per sub-index |
| | ATBREAK: Index contents separated into separate files at each <BDE_BREAK> marker (default: SINGLE) |
| HEAD | Header template to use for generated files (default: GXEngine's HeaderTemplate property) |
| FOOT | Footer template to use for generated files (default: GXEngine's FooterTemplate property) |

### Template input

```
Customers by letter:<BR>
<BDE_INDEX NAME="Index1" FILES=ATBREAK>
<BDE_GROUP NAME="FirstChar" SRC="orderlist" FLDS="Name" CHARS=1>
Customers from <BDE_FIELD SRC="FirstChar" FLDS="LOW"> to <BDE_FIELD SRC="FirstChar" FLDS="HIGH">
<BR>
<BDE_SCAN SRC="orderlist">
Customer <B><BDE_FIELD SRC="orderlist" FLDS="Name" INDEX="Name"></B><BR>
</BDE_SCAN>
<BDE_BREAK>
</BDE_GROUP>
</BDE_INDEX>
```

### HTML output

```
Customers by letter:<BR>
<UL>
<LI><A HREF=”sam8101.HTM#Index1=Abacus”=Abacus - Azziz</A>
<LI><A HREF=”sam8101.HTM#Index1=Bentley”=Bentley - Bjork</A>
<LI><A HREF=”sam8101.HTM#Index1=Costus”>Costus - Crow</A>
...
</UL>
<BR>
```

### Web browser output

```
Customers by letter:
Abacus-Azziz
Bentley-Bjork
Costus-Crow
```

...

### HTML output of sam8101.HTM

```
Customers from Abacus to Azziz<BR>
Customer <B><A NAME=”Index1=Abacus”></A>Abacus</B><BR>
Customer <B>Arton</B><BR>
Customer <B>Azziz</B><BR>
Customers from Bentley to Bjork<BR>
Customer <B><A NAME=”Index1=Bentley”></A>
Customer <B>Bentley</B><BR>
...
```

### Web browser output of sam8101.HTM

Customers from Abacus to Azziz
Customer **Abacus**
Customer **Arton**
Customer **Azziz**
Customers from Bentley to Bjork

Customer **Bentley**
...

---

**{button ,AL(`GXTEMPSPEC;GXTOKEN;',0,"Defaultoverview",)}** **Related Topics**

# BDE_BREAK (HTML template tag)

Marks a segment break for an Index (Table of Contents) construct.

## Syntax

```
<BDE_BREAK>
```

## User examples

```
<BDE_INDEX TEMPLATE=”ORDERSLIST”>
<BDE_SCAN SRC=”CUSTOMER”>
<BDE_TABLE SRC=”ORDERS” FLDS=”OrderNo” INDEX=”OrderNo”>
<BDE_BREAK>
</BDE_SCAN>
</BDE_INDEX>
```

---

**{button ,AL(`GXTEMPSPEC;GXTOKEN;',0,"Defaultoverview",)}** **Related Topics**

# Data fields in the special "INDEX" *sourceref*

"INDEX" is a reserved data source that contains several useful field values.

### Syntax

```
<BDE_FIELD SRC=sourceref FLDS=indexfield>
```

### Parameters

`SRC` Must be set to "Index" (not case-sensitive)

`FLDS` Can be one or more of

LOW: First value to appear in index range

HIGH: Last value to appear in index range

RANGE: Combined first and last values separated by dash (-)

HREF: HTML-formatted reference string defining target document and position within that document in accordance with the standard HTML <A HREF> tag

### Sample document template

```
<BDE_INDEX NAME=”CustIndex” TEMPLATE=”CUSTLIST”>
<BDE_SCAN SRC=”CUSTOMER”>
Customer Number: <BDE_FIELD FLDS=”Name” INDEX=”Name”>
<BDE_SCAN SRC=”ORDERS”>
Order Number: <BDE_FIELD FLDS=”OrderNo” INDEX=”OrderNo”>
</BDE_SCAN>
<BDE_BREAK>
</BDE_SCAN>
</BDE_INDEX>
```

### Sample INDEX template

```
<UL>
<BDE_SCAN SRC=”INDEX”>
<LI><A HREF=”<BDE_FIELD FLDS=”HREF”>”>
  <BDE_FIELD FLDS=”LOW”>
  to
  <BDE_FIELD FLDS=”HIGH”></A>
</BDE_SCAN>
</UL>
```

---

**{button ,AL(`GXTEMPSPEC;GXTOKEN;',0,"Defaultoverview",)} Related Topics**

## Examples: BDE_INDEX usage

The next several topics illustrate the use of indexes, starting with a non-indexed template, and adding indexes as needed. New template code appears in bold for each iteration.

- BDE_INDEX Example: Non-indexed template
- BDE_INDEX Example: Non-indexed Web browser output
- BDE_INDEX Example: Template with an index on the Name field
- BDE_INDEX Example: Web browser output with an index on the Name field
- BDE_INDEX Example: Template with another index on the OrderNo field
- BDE_INDEX Example: Web browser output with another index on the OrderNo field
- BDE_INDEX Example: Template with bulleted customer order lists
- BDE_INDEX Example: Web browser output with bulleted customer order lists

---

**{button ,AL(`GXTEMPSPEC;GXTOKEN;',0,"Defaultoverview",)}** **Related Topics**

# BDE_INDEX Example: Non-indexed template

BDE_INDEX is not used in this template. For an example of its browser output, see BDE_INDEX Example: Non-indexed Web browser output.

### Template example

```
<BDE_GROUP NAME=”CustGroup” SRC=”CUSTOMER” FLDS=”Name” CHARS=1 INCR=5>
<B><FONT SIZE=6>Customers <BDE_FIELD SRC=”CustGroup” FLDS=”LOW”> to <BDE_FIELD SRC=”CustGroup”
FLDS=”HIGH”></FONT></B><BR>
<BDE_SCAN SRC=”CUSTOMER”>
<B>Customer: <BDE_FIELD FLDS=”Name”></B><BR>
<BDE_SCAN SRC=”ORDERS”>
Order Number: <BDE_FIELD FLDS=”OrderNo”><BR>
<BDE_TABLE SRC=”LINEITEM” BORDER><BR>
</BDE_SCAN>
</BDE_SCAN>
</BDE_GROUP>
```

---

**{button ,AL(`GXTEMPSPEC;GXTOKEN;',0,"Defaultoverview",)} Related Topics**

# BDE_INDEX Example: Non-indexed Web browser output

After the template in BDE_INDEX Example: Non-indexed template is run through the GXEngine, it looks like the following in a Web browser:

### Output example

**Customers Alfa to Eggbert**

**Customer: Alfa**

Order Number: 1000

| OrderNo | ItemNo | Qty |
|---|---|---|
| 1000 | 122 | 1 |
| 1000 | 123 | 3 |

Order Number: 1001

| OrderNo | ItemNo | Qty |
|---|---|---|
| 1001 | 110 | 2 |

**Customer: Eggbert**

Order Number: 3999

| OrderNo | ItemNo | Qty |
|---|---|---|
| 3999 | 313 | 5 |

Order Number: 4000

| OrderNo | ItemNo | Qty |
|---|---|---|
| 4000 | 314 | 3 |

**Customers Fleecer to Juckiera**

**Customer: Fleecer**

Order Number: 1000

| OrderNo | ItemNo | Qty |
|---|---|---|
| 3993 | 122 | 11 |

Order Number: 1001

| OrderNo | ItemNo | Qty |
|---|---|---|
| 4002 | 110 | 11 |
| 5901 | 111 | 66 |

...

---

**{button ,AL(`GXTEMPSPEC;GXTOKEN;',0,"Defaultoverview",)} Related Topics**

# BDE_INDEX Example: Template with an index on the Name field

This example adds an index to the template shown in BDE_INDEX Example: Non-indexed template.

To make the entire document a bulleted list, the BDE_INDEX construct is put on the outside of the outermost group. Bold type highlights additions to the template. To see how this template appears when processed by the GXEngine and viewed through a Web browser, see BDE_INDEX Example: Web browser output with an index on the Name field.

### Template example

```
<BDE_INDEX NAME="CUSTINDEX" FILES=ATBREAK>
<BDE_GROUP NAME="CustGroup" SRC="CUSTOMER" FLDS="Name" CHARS=1 INCR=5>
<B><FONT SIZE=6>Customers <BDE_FIELD SRC="CustGroup" FLDS="LOW"> to <BDE_FIELD SRC="CustGroup"
FLDS="HIGH"></FONT></B><BR>
<BDE_SCAN SRC="CUSTOMER">
<B>Customer: <BDE_FIELD FLDS="Name" INDEX="Name"></B><BR>
<BDE_SCAN SRC="ORDERS">
Order Number: <BDE_FIELD FLDS="OrderNo"><BR>
<BDE_TABLE SRC="LINEITEM"><BR>
</BDE_SCAN>
</BDE_SCAN>
<BDE_BREAK>
</BDE_GROUP>
</BDE_INDEX>
```

---

**{button ,AL(`GXTEMPSPEC;GXTOKEN;',0,"Defaultoverview",)} Related Topics**

# BDE_INDEX Example: Web browser output with an index on the Name field

After the template in BDE_INDEX Example: Template with an index on the Name field is run through the GXEngine, it looks like the following in a Web browser:

### Output example

Each of these items is a link to a new document:

- **Alfa - Eggbert**
- **Fleecer - Juckiera**

...

Example of output after clicking on the "Alfa-Eggbert" link in a Web browser:

**Customers Alfa to Eggbert**

**Customer: Alfa**

Order Number: 1000

| OrderNo | ItemNo | Qty |
|---|---|---|
| 1000 | 122 | 1 |
| 1000 | 123 | 3 |

Order Number: 1001

| OrderNo | ItemNo | Qty |
|---|---|---|
| 1001 | 110 | 2 |

**Customer: Eggbert**

Order Number: 3999

| OrderNo | ItemNo | Qty |
|---|---|---|
| 3999 | 313 | 5 |

Order Number: 4000

| OrderNo | ItemNo | Qty |
|---|---|---|
| 4000 | 314 | 3 |

---

**{button ,AL(`GXTEMPSPEC;GXTOKEN;',0,"Defaultoverview",)} Related Topics**

# BDE_INDEX Example: Template with another index on the OrderNo field

This example adds another indexed field ("OrderNo") to the template shown in BDE_INDEX Example: Template with an index on the Name field. Now, order numbers appear in the index links. Bold type highlights template additions.

To see how this template appears when processed by the GXEngine and viewed through a Web browser, see BDE_INDEX Example: Web browser output with another index on the OrderNo field.

### Template example

```
<BDE_INDEX NAME=”CUSTINDEX” FILES=ATBREAK>
<BDE_GROUP NAME=”CustGroup” SRC=”CUSTOMER” FLDS=”Name” CHARS=1 INCR=5>
<B><FONT SIZE=6>Customers <BDE_FIELD SRC=”CustGroup” FLDS=”LOW”> to <BDE_FIELD SRC=”CustGroup”
   FLDS=”HIGH”></FONT></B><BR>
<BDE_SCAN SRC=”CUSTOMER”>
<B>Customer: <BDE_FIELD FLDS=”Name” INDEX=”Name”></B><BR>
<BDE_SCAN SRC=”ORDERS”>
Order Number: <BDE_FIELD FLDS=”OrderNo” INDEX=”OrderNo”><BR>
<BDE_TABLE SRC=”LINEITEM”><BR>
</BDE_SCAN>
</BDE_SCAN>
<BDE_BREAK>
</BDE_GROUP>
</BDE_INDEX>
```

---

**{button ,AL(`GXTEMPSPEC;GXTOKEN;',0,"Defaultoverview",)} Related Topics**

# BDE_INDEX Example: Web browser output with another index on the OrderNo field

After the template in BDE_INDEX Example: Template with another index on the OrderNo field is run through the GXEngine, it looks like the following output example in a Web browser.

Notice that the first and last order numbers appear along with the first and last customer name within the index.

You can customize the output and formatting of these links by adding your own index template to the GXEngine, then setting the GXEngine's IndexTemplate property to the name you specify.

## Output example

- **Alfa,1000 - Eggbert,4000**
- **Fleecer,1000 - Juckiera,7894**

...

---

**{button ,AL(`GXTEMPSPEC;GXTOKEN;',0,"Defaultoverview",)} Related Topics**

# BDE_INDEX Example: Template with bulleted customer order lists

This example modifies the template shown in BDE_INDEX Example: Template with another index on the OrderNo field to display the orders for each customer as a bulleted list. Bold type highlights additions to the template.

To see how this template appears when processed by the GXEngine and viewed through a Web browser, see BDE_INDEX Example: Web browser output with bulleted customer order lists.

## Template example

```
<BDE_INDEX NAME=”CUSTINDEX” FILES=ATBREAK>
<BDE_GROUP NAME=”CustGroup” SRC=”CUSTOMER” FLDS=”Name” CHARS=1 INCR=5>
<B><FONT SIZE=6>Customers <BDE_FIELD SRC=”CustGroup” FLDS=”LOW”> to <BDE_FIELD SRC=”CustGroup”
   FLDS=”HIGH”></FONT></B><BR>
<BDE_SCAN SRC=”CUSTOMER”>
<B>Customer: <BDE_FIELD FLDS=”Name” INDEX=”Name”></B><BR>
<BDE_INDEX NAME=”ORDINDEX” FILES=ATBREAK>
<BDE_SCAN SRC=”ORDERS”>
Order Number: <BDE_FIELD FLDS=”OrderNo” INDEX=”OrderNo”><BR>
<BDE_TABLE SRC=”LINEITEM”><BR>
<BDE_BREAK>
</BDE_SCAN>
</BDE_INDEX>
</BDE_SCAN>
<BDE_BREAK>
</BDE_GROUP>
</BDE_INDEX>
```

---

**{button ,AL(`GXTEMPSPEC;GXTOKEN;',0,"Defaultoverview",)} Related Topics**

# BDE_INDEX Example: Web browser output with bulleted customer order lists

After the template in BDE_INDEX Example: Template with bulleted customer order lists is run through the GXEngine, it looks like the following in a Web browser.

Note that the extra OrderNo index field no longer appears in the "outside" index.

## Template output

- **Alfa - Eggbert**
- **Fleecer - Juckiera**

...

Example of output after clicking on the "Alfa-Eggbert" link in a Web browser:

**Customers Alfa to Eggbert**

**Orders for Customer: Alfa**

- **1000**
- **1001**

**Orders for Customer: Eggbert**

- **3999**
- **4000**

Example of output after clicking on the "1000" link in a Web browser:

Order Number: 1000

| OrderNo | ItemNo | Qty |
|---|---|---|
| 1000 | 122 | 1 |
| 1000 | 123 | 3 |

---

**{button ,AL(`GXTEMPSPEC;GXTOKEN;',0,"Defaultoverview",)} Related Topics**

# BDE_INCLUDE (HTML template tag)

Inserts another template at the current position. Any included templates must be added to the GXEngine prior to executing the template that contains the reference. (See the GXEngine Interface methods AddTemplateFile and AddTemplateString.)

### Syntax

```
<BDE_INCLUDE TEMPLATE=templateref>
```

### Parameters

| | |
|---|---|
| TEMPLATE | Template name to insert |

### User examples

```
<BDE_INCLUDE TEMPLATE=”ENTERCUSTOMER”>
<BDE_INCLUDE TEMPLATE=”ORDERINFO”>
```

---

**{button ,AL(`GXTEMPSPEC;GXTOKEN;',0,"Defaultoverview",)} Related Topics**

# GXEngine Interface Specification

The GXEngine is an OLE Automation server, used to convert templates into static HTML documents (see GXEngine DLL and The GXEngine Template Specification for more information on the GXEngine and templates).

You can use the methods and properties listed in the following topics to perform template-to-HTML conversions using the GXEngine. They can be called using the usual techniques for your language:

- GXEngine Interface methods
- GXEngine Interface properties

In typical use, the templates, data sources, and parameters involved in a conversion are first added to an instance of the GXEngine OLE Automation server, then output is generated by "executing" that GXEngine instance (calling the Execute or ExecuteFrom method).

First, an instance of the GXEngine is created by invoking your language's method for creating an OLE Automation object and passing it the server name `"Corel.GXEngine"`.

For a typical code sample, see GXEngine Interface ObjectPAL code sample.

The GXEngine was designed to cache any added data sources during the lifetime of the GXEngine object, so performance typically improves with repeated use.

---

**{button ,AL(`GXENGINE;GXINTERSPEC;',0,"Defaultoverview",)} Related Topics**

# GXEngine Interface ObjectPAL code sample

The following script starts the GXEngine, converts the VENDORS.DB table in the working directory to HTML, and saves it as the file VENDORS.HTM in the working directory:

```
method run(var eventInfo Event)
var
  GXEngine OLEAuto
endVar
  GXEngine.Open ("Corel.GXEngine")
  GXEngine^AddTemplateString ("vendors","<BDE_TABLE SRC=\"VENDORS\" BORDER=2>")
  GXEngine^AddSourceTable ("vendors",fullName(":WORK:"),"vendors.db")
  GXEngine^OutputMethod = 1  ;output to a file
  GXEngine^OutputFileName = "vendors.htm"   ;output to the working directory
  GXEngine^Execute()
endMethod
```

In the AddTemplateString and AddSourceTable parameters, `vendors` is the identifier that is assigned to VENDORS.DB, similar to an alias. The second parameter for AddSourceTable is *`database`*, which gives the location of the source table for the template. It can be expressed as

- a standard alias
- an SQL alias
- WORK or PRIV aliases, expressed as shown in the example, with fullName used to return the complete path for the working directory
- the fully qualified path to the location of the source table, for example:

  `"C:\\COREL\\SUITE8\\PARADOX\\SAMPLE"`

Notice that the caret (^) is used as a delimiter for all the OLE Automation methods and properties except Open. Corel Paradox recognizes the Open method as an oleAuto method, but it doesn't necessarily recognize other methods and properties of the GXEngine and may execute them incorrectly when encountered in a script.

---

**{button ,AL(`GXINTERSPEC;GXMETHOD_INTRO;GXPROPERTY_INTRO;',0,"Defaultoverview",)} Related Topics**

# GXEngine Interface methods

The following methods are used to control the GXEngine DLL:

| | |
|---|---|
| Execute | Generates HTML output using the current templates, data sources, properties, and parameters |
| ExecuteFrom | Generates HTML output from a specific position using the current templates, data sources, properties, and parameters |
| AddSourceTable | Adds a data source from a table in a directory or BDE alias |
| RemoveSource | Removes a data source from the GXEngine |
| RefreshSources | Updates record buffers in all open data sources |
| AddLink | Adds a link between two data sources |
| RemoveLink | Removes a link between two data sources |
| AddTemplateFile | Adds a template from a file to the GXEngine |
| AddTemplateString | Adds a template from a string to the GXEngine |
| RemoveTemplate | Removes a template from the GXEngine |
| AddParam | Adds a parameter to the GXEngine |
| RemoveParam | Removes a parameter from the GXEngine |

---

**{button ,AL(`GXINTERSPEC;',0,"Defaultoverview",)} Related Topics**

# GXEngine Interface properties

You can set the following properties for the GXEngine DLL:

| | |
|---|---|
| MasterTemplate | The name of the template to process first |
| IndexTemplate | The name of the default template to use for the BDE_INDEX template tag's *template* parameter |
| HeaderTemplate | The name of the default template to use for the BDE_INDEX template tag's *head* parameter |
| FooterTemplate | The name of the default template to use for the BDE_INDEX template tag's *foot* parameter |
| OutputMethod | Specifies whether the conversion should output to a file (1) or string (2) |
| OutputFileName | Specifies the name and destination for the output file; used only if OutputMethod = 1 |
| OutputString | Contains the HTML output generated by calling Execute or ExecuteFrom. Used only if OutputMethod = 2 |
| OutputPath | The default location for storing output; the default OutputPath = the path to OutputFileName |
| ImagePath | The default location for storing binary image output; the default ImagePath = OutputPath |
| PrivateDir | Directory to use for all temporary or auxiliary files; the default = Windows temporary directory |
| LongFileNames | Specifies whether the directory in OutputFileName supports long filenames |
| Dynamic | Indicates whether the document to be published is dynamic or static |
| NumberFormat | Specifies the default format for all integer conversions |
| FloatFormat | Specifies the default format for all floating-point conversions |
| CurrencyFormat | Specifies the default format for all currency conversions |
| DateFormat | Specifies the default format for all date conversions |
| TimeFormat | Specifies the default format for all time conversions |
| DateTimeFormat | Specifies the default format for all date-time conversions |

---

**{button ,AL(`GXINTERSPEC;',0,"Defaultoverview",)}** **Related Topics**

# Execute (GXEngine method)

Generates HTML output using the current templates, data sources, properties, and parameters.

## Syntax

```
Execute()
```

For an example, see GXEngine Interface ObjectPAL code sample.

---

**{button ,AL(`GXINTERSPEC;GXMETHOD;',0,"Defaultoverview",)} Related Topics**

# ExecuteFrom (GXEngine method)

Generates HTML output from a specific position using the current templates, data sources, properties, and parameters.

## Syntax

```
ExecuteFrom(const QueryString String)
```

## Description

This method uses QueryString, the portion of an HTTP statement to the right of the ?, to indicate what should be converted to HTML. This method is used with BDE_INDEX.

Unlike the Execute( ) method, which generates output beginning at the top of the template, ExecuteFrom( ) takes a QueryString that specifies the index from which the GXEngine will begin output. This is how the GXPublish component implements dynamic drill-down reporting.

The format for the QueryString is as follows:

"*Indexref = Value&Indexref=Value&...*"

for example,

"CUSTINDEX=Alfa&ORDINDEX=1001"

---

**{button ,AL(`GXINTERSPEC;GXMETHOD;',0,"Defaultoverview",)} Related Topics**

# AddSourceTable (GXEngine Interface method)

Adds a data source from a table in a directory or BDE alias to the GXEngine. For an example, see GXEngine Interface ObjectPAL code sample.

### Syntax

```
AddSourceTable(const SourceRef String, const DatabaseName String, const TableName String)
```

### Description

SourceRef is an identifier for the TableName located at DatabaseName.

DatabaseName is the location of the table whose data is to be used in a template. It can be expressed as

- a standard alias
- an SQL alias
- WORK or PRIV aliases, expressed as shown in the example, with fullName used to return the complete path for the working directory
- the fully qualified path to the location of the source table, for example:

  ```
  "C:\\COREL\\SUITE8\\PARADOX\\SAMPLE"
  ```

TableName is the name of a Corel Paradox, dBASE, or SQL table to be represented by SourceRef.

---

**{button ,AL(`GXINTERSPEC;GXMETHOD;',0,"Defaultoverview",)} Related Topics**

# RemoveSource (GXEngine Interface method)

Removes a data source from the GXEngine.

### Syntax

```
RemoveSource(const SourceRef String)
```

### Description

SourceRef is the identifier of the data source, previously added with AddSourceTable, to remove.

---

**{button ,AL(`GXINTERSPEC;GXMETHOD;',0,"Defaultoverview",)} Related Topics**

# RefreshSources (GXEngine Interface method)

Updates record buffers in all open data sources.

## Syntax

```
RefreshSources( )
```

## Description

It is necessary to call this method only for an instance of the GXEngine whose data sources have changed when an up-to-date view is required.

---

**{button ,AL(`GXINTERSPEC;GXMETHOD;',0,"Defaultoverview",)}** **Related Topics**

# AddLink (GXEngine Interface method)

Adds a link between two data sources.

### Syntax

```
AddLink(const MasterSource String, const DetailSource String, const MasterFlds String, const DetailFlds String)
```

### Description

Links the master data source to a detail data source (where both data sources were previously added with AddSourceTable). MasterFlds is a comma-separated list of fields to use from the master table and DetailsFlds is a similar list from the detail table.

This method is useful in maintaining one-to-many relationships when iterating data sources in the template. For example, calling `AddLink("Customers","Orders","CustNo","CustNo")` tells the GXEngine to show only those records in Orders where the CustNo field matches the current CustNo field in the Customer table.

**Note**

- The same DetailSource cannot be used in two separate links. For example, if, after performing the above statement, you execute this:
  `AddLink("OldCustomers","Orders","CustNo","CustNo")`

  Then Orders would be linked to OldCustomers, not Customers, as it was previously.

---

**{button ,AL(`GXINTERSPEC;GXMETHOD;',0,"Defaultoverview",)} Related Topics**

# RemoveLink (GXEngine Interface method)

Removes a link between two data sources.

### Syntax

```
RemoveLink(const DetailSource string)
```

### Description

DetailSource is the detail data source (added with AddSourceTable) which uniquely identifies the link (added with AddLink).

---

**{button ,AL(`GXINTERSPEC;GXMETHOD;',0,"Defaultoverview",)} Related Topics**

# AddTemplateFile (GXEngine Interface method)

Adds a template from a text file to the GXEngine.

## Syntax

```
AddTemplateFile(const TemplateRef String, const InputFile String)
```

## Description

Loads the contents of InputFile and assigns it to the template identifier TemplateRef.

---

**{button ,AL(`GXINTERSPEC;GXMETHOD;',0,"Defaultoverview",)} Related Topics**

# AddTemplateString (GXEngine Interface method)

Adds a template from a string to the GXEngine.

## Syntax

```
AddTemplateString(const TemplateRef String, const InputString String)
```

## Description

Assigns the contents of the string InputString to the template identifier TemplateRef.

---

**{button ,AL(`GXINTERSPEC;GXMETHOD;',0,"Defaultoverview",)} Related Topics**

# RemoveTemplate (GXEngine Interface method)

Removes a template from the GXEngine.

## Syntax

```
RemoveTemplate(const TemplateRef String)
```

## Description

Removes the template identified by TemplateRef (added with AddTemplateFile or AddTemplateString) from the GXEngine.

---

**{button ,AL(`GXINTERSPEC;GXMETHOD;',0,"Defaultoverview",)} Related Topics**

# AddParam (GXEngine Interface method)

Adds a parameter to the GXEngine.

### Syntax

```
AddParam(const ParamRef String, const ParamValue String)
```

### Description

Assigns the parameter identifier ParamRef to ParamValue. This creates a new parameter for use with the BDE_FIELD template tag.

---

**{button ,AL(`GXINTERSPEC;GXMETHOD;',0,"Defaultoverview",)} Related Topics**

# RemoveParam (GXEngine Interface method)

Removes a parameter from the GXEngine.

## Syntax

```
RemoveParam(const ParamRef String)
```

## Description

Removes the parameter identified by ParamRef.

---

**{button ,AL(`GXINTERSPEC;GXMETHOD;',0,"Defaultoverview",)} Related Topics**

# IndexTemplate (GXEngine Interface property)

### Data type

String

### Description

The name of the default template to use for the BDE_INDEX template tag's *template* parameter.

The template used must be added using AddTemplateFile or AddTemplateString before the Execute( ) or ExecuteFrom( ) method is called.

### Value

N/A

---

**{button ,AL(`GXINTERSPEC;GXPROPERTY;',0,"Defaultoverview",)} Related Topics**

# MasterTemplate (GXEngine Interface property)

### Data type

String

### Description

The name of the template to control the conversion. Other templates may eventually become involved in the conversion. (See BDE_INCLUDE.)

### Value

N/A

---

**{button ,AL(`GXINTERSPEC;GXPROPERTY;',0,"Defaultoverview",)} Related Topics**

# HeaderTemplate (GXEngine Interface property)

### Data type

String

### Description

The name of the default template to use for the BDE_INDEX template tag's *head* parameter.

### Value

N/A

---

**{button ,AL(`GXINTERSPEC;GXPROPERTY;',0,"Defaultoverview",)}** **Related Topics**

# FooterTemplate (GXEngine Interface property)

### Data type

String

### Description

The name of the default template to use for the BDE_INDEX template tag's *foot* parameter.

### Value

N/A

---

**{button ,AL(`GXINTERSPEC;GXPROPERTY;',0,"Defaultoverview",)}** **Related Topics**

# OutputFileName (GXEngine Interface property)

## Data type

String

## Description

Specifies the name and destination for the output file.

If multiple files are created (image files or files resulting from BDE_INDEX), then they are created in the OutputPath directory or ImagePath directory.

## Value

N/A

---

**{button ,AL(`GXINTERSPEC;GXPROPERTY;',0,"Defaultoverview",)} Related Topics**

# OutputString (GXEngine Interface property)

### Data type

String

### Description

Contains the HTML output after calling Execute or ExecuteFrom; used only if OutputMethod = 2.
See OutputFileName for information about multiple files.

### Value

N/A

---

**{button ,AL(`GXINTERSPEC;GXPROPERTY;',0,"Defaultoverview",)} Related Topics**

# OutputPath (GXEngine Interface property)

### Data type

String

### Description

Specifies the default location for storing HTML output, the default for OutputPath is the path to OutputFileName.

If a destination path is included in OutputFileName, that path is used for a single file. See OutputFileName for information about multiple files.

### Value

N/A

---

**{button ,AL(`GXINTERSPEC;GXPROPERTY;',0,"Defaultoverview",)} Related Topics**

# ImagePath (GXEngine Interface property)

### Data type

String

### Description

The default location for storing binary image output; the default ImagePath is OutputPath.

### Value

N/A

---

**{button ,AL(`GXINTERSPEC;GXPROPERTY;',0,"Defaultoverview",)} Related Topics**

# PrivateDir (GXEngine Interface property)

### Data type

String

### Description

The directory to use for all temporary or auxiliary files; the default is the Windows temporary directory.

### Value

N/A

---

**{button ,AL(`GXINTERSPEC;GXPROPERTY;',0,"Defaultoverview",)}** **Related Topics**

# OutputMethod (GXEngine Interface property)

### Data type

Int

### Description

Specifies whether the conversion should output to a file (1) or string (2).

### Value

1 or 2 (default)

---

**{button ,AL(`GXINTERSPEC;GXPROPERTY;',0,"Defaultoverview",)}** **Related Topics**

# LongFileNames (GXEngine Interface property)

## Data type

Boolean

## Description

Specifies whether the directory in OutputFileName supports long filenames.

If set to TRUE, multiple images and files created during the conversion will be named by the OutputFileName method (excluding any extension) with an appended series of underscores ( _ ) and numbers to indicate its nesting level and sequence—for example, `OUTPUT_2_3.HTM`.

If set to FALSE, the GXEngine uses the Win32 function GetTempFileName( ) to guarantee a unique filename that is restricted to the 8-character name and 3-character extension.

## Value

TRUE or FALSE (default)

---

**{button ,AL(`GXINTERSPEC;GXPROPERTY;',0,"Defaultoverview",)} Related Topics**

# Dynamic (GXEngine Interface property)

### Data type

Boolean

### Description

Specifies whether the HTML output being processed is dynamic (TRUE) or static (FALSE).

### Value

TRUE or FALSE

---

**{button ,AL(`GXINTERSPEC;GXPROPERTY;',0,"Defaultoverview",)}** **Related Topics**

# NumberFormat (GXEngine Interface property)

### Data type

String

### Description

Specifies the default format for all integer conversions.

### Value

Number format values

Default = Windows locale settings.

---

**{button ,AL(`GXINTERSPEC;GXPROPERTY;',0,"Defaultoverview",)} Related Topics**

# FloatFormat (GXEngine Interface property)

### Data type

String

### Description

Specifies the default format for all floating-point conversions.

### Value

Number format values

Default = Windows locale settings.

---

**{button ,AL(`GXINTERSPEC;GXPROPERTY;',0,"Defaultoverview",)} Related Topics**

# CurrencyFormat (GXEngine Interface property)

### Data type

String

### Description

Specifies the default format for all currency conversions.

### Value

Number format values

Default = Windows locale settings.

---

**{button ,AL(`GXINTERSPEC;GXPROPERTY;',0,"Defaultoverview",)} Related Topics**

# DateFormat (GXEngine Interface property)

### Data type

String

### Description

Specifies the default format for all date conversions.

### Value

Date and time format values

Default = Windows locale settings.

---

**{button ,AL(`GXINTERSPEC;GXPROPERTY;',0,"Defaultoverview",)}** **Related Topics**

# TimeFormat (GXEngine Interface property)

### Data type

String

### Description

Specifies the default format for all time conversions.

### Value

Date and time format values

Default = Windows locale settings.

---

**{button ,AL(`GXINTERSPEC;GXPROPERTY;',0,"Defaultoverview",)} Related Topics**

# DateTimeFormat (GXEngine Interface property)

### Data type

String

### Description

Specifies the default format for all date-time conversions.

### Value

Date and time format values

Default = Windows locale settings.

---

**{button ,AL(`GXINTERSPEC;GXPROPERTY;',0,"Defaultoverview",)} Related Topics**

# Number format values

| Specifier | Represents |
|---|---|
| 0 | Digit placeholder. If the value being formatted has a digit in the position where the '0' appears in the format string, then that digit is copied to the output string. Otherwise, a '0' is stored in that position in the output string. |
| # | Digit placeholder. If the value being formatted has a digit in the position where the '#' appears in the format string, then that digit is copied to the output string. Otherwise, nothing is stored in that position in the output string. |
| . | Decimal point. The first '.' character in the format string determines the location of the decimal separator in the formatted value; any additional '.' characters are ignored. The actual character used as the decimal separator in the output string is determined by the Number Format of the Regional Settings in the Windows Control Panel. |
| , | Thousand separator. If the format string contains one or more ',' characters, the output will have thousand separators inserted between each group of three digits to the left of the decimal point. The placement and number of ',' characters in the format string does not affect the output, except to indicate that thousand separators are wanted. The actual character used as the thousand separator in the output is determined by the Number Format of the Regional Settings in the Windows Control Panel. |
| E+ | Scientific notation. If any of the strings 'E+', 'E-', 'e+', or 'e-' are contained in the format string, the number is formatted using scientific notation. A group of up to four '0' characters can immediately follow the 'E+', 'E-', 'e+', or 'e-' to determine the minimum number of digits in the exponent. The 'E+' and 'e+' formats cause a plus sign to be output for positive exponents and a minus sign to be output for negative exponents. The 'E-' and 'e-' formats output a sign character only for negative exponents. |
| 'xx'/"xx" | Characters enclosed in single or double quotes are output as is and do not affect formatting. |
| ; | Separates sections for positive, negative, and zero numbers in the format string. |

The locations of the leftmost '0' before the decimal point in the format string and the rightmost '0' after the decimal point in the format string determine the range of digits that are always present in the output string.

The number being formatted is always rounded to as many decimal places as there are digit placeholders ('0' or '#') to the right of the decimal point. If the format string contains no decimal point, the value being formatted is rounded to the nearest whole number.

If the number being formatted has more digits to the left of the decimal separator than there are digit placeholders to the left of the '.' character in the format string, the extra digits are output before the first digit placeholder.

To allow different formats for positive, negative, and zero values, the format string can contain between one and three sections separated by semicolons:

- One section: The format string applies to all values.
- Two sections: The first section applies to positive values and zeros, and the second section applies to negative values.
- Three sections: The first section applies to positive values, the second applies to negative values, and the third applies to zeros.

If the section for negative values or the section for zero values is empty, that is, if there is nothing between the semicolons that delimit the section, the section for positive values is used instead.

# Date and time format values

DateFormat, TimeFormat, and DateTimeFormat GXEngine Interface properties format date and time values using the appropriate parts of a string composed from the following format specifiers:

| Specifier | Displays |
|---|---|
| c | Displays the date using the format given by the Date Format of the Regional Settings in the Windows Control Panel followed by the time with seconds. The time is not displayed if the fractional part of the DateTime value is zero. |
| d | Displays the day as a number without a leading zero (1-31) |
| dd | Displays the day as a number with a leading zero (01-31) |
| ddd | Displays the day as an abbreviation (Sun-Sat) |
| dddd | Displays the day as a full name (Sunday-Saturday) |
| ddddd | Displays the date using the format given by the Date Format of the Regional Settings in the Windows Control Panel |
| dddddd | Displays the date using the format given by the Date Format of the Regional Settings in the Windows Control Panel |
| m | Displays the month as a number without a leading zero (1-12). If the m specifier immediately follows an h or hh specifier, the minute rather than the month is displayed |
| mm | Displays the month as a number with a leading zero (01-12). If the mm specifier immediately follows an h or hh specifier, the minute rather than the month is displayed |
| mmm | Displays the month as an abbreviation (Jan-Dec) |
| mmmm | Displays the month as a full name (January-December) |
| yy | Displays the year as a two-digit number (00-99) |
| yyyy | Displays the year as a four-digit number (0000-9999) |
| h | Displays the hour without a leading zero (0-23) |
| hh | Displays the hour with a leading zero (00-23) |
| n | Displays the minute without a leading zero (0-59) |
| nn | Displays the minute with a leading zero (00-59) |
| s | Displays the second without a leading zero (0-59) |
| ss | Displays the second with a leading zero (00-59) |
| t | Displays the time without seconds |
| tt | Displays the time with seconds |
| am/pm | Uses the 12-hour clock for the preceding h or hh specifier, and displays 'am' for any hour before noon, and 'pm' for any hour after noon. The am/pm specifier can use lower, upper, or mixed case, and the result is displayed accordingly. |
| a/p | Uses the 12-hour clock for the preceding h or hh specifier, and displays 'a' for any hour before noon, and 'p' for any hour after noon. The a/p specifier can use lower, upper, or mixed case, and the result is displayed accordingly. |
| ampm | Uses the 12-hour clock for the preceding h or hh specifier, and displays the contents of the Time Format of the Regional Settings in the Windows Control Panel for any hour before or after noon |
| / | Displays the date separator character given by the Date Format of the Regional Settings in the Windows Control Panel |
| : | Displays the time separator character given by the Time Format of the Regional Settings in the Windows Control Panel |
| 'xx'/"xx" | Characters enclosed in single or double quotes are displayed as is, and do not affect formatting |

Format specifiers may be written in upper-case as well as in lower-case letters—both produce the same result. If the string given by the Format parameter is empty, the date and time value is formatted as if a 'c' format specifier had been given.

# About the GXPublish DLL

The GXPublish DLL is an OLE Automation server that serves as an HTTP request handler registered in the Web Server Repository. This is the component that knows how to produce HTML output dynamically from Corel Paradox documents. Publishing features include

- **Table views**—The HTML Table Expert uses the GXPublish component to publish a table view.
- **Reports**—The HTML Report Expert to outputs static HTML documents or HTML report templates. When any Web browser requests a dynamic HTML report template, the GXPublish component automatically generates the HTML report
—filling in a snapshot of data at the time of the request.

---

**{button ,AL(`OVERVIEW;',0,"Defaultoverview",)} Related Topics**

# HTMLIB01 Corel Paradox HTML Publishing Library Specification

The Corel Paradox HTML Publishing Library is an ObjectPAL library that provides an interface to both the static and dynamic HTML publishing features in Paradox. To access the library, get the Experts directory using the ExpertDir() procedure, then load the HTMLIB01 library.

The library includes

- Types
- Methods

---

**{button ,AL(`HTMLIBSPEC;',0,"Defaultoverview",)}** **Related Topics**

# HTMLIB01 Corel Paradox HTML Publishing Library types

There are several predefined types in the Corel Paradox HTML Publishing Library. These types are used as parameters for several of the methods in the library.

| | | | |
|---|---|---|---|
| _arString | Array[] String | | |
| _arLongInt | Array[] LongInt | | |
| _arBinary | Array[] Binary | | |
| _dynString | DynArray[] String | | |
| _dynMemo | DynArray[] Memo | | |
| _dynTCursor | DynArray[] Tcursor | | |
| _dynSmallInt | DynArray[] SmallInt | | |
| _dynBinary | DynArray[] Binary | | |
| _HTMLTable | strURI | String | URI (or output filename) |
| (record) | strTableName | String | Source table name |
| | strSource | String | Data source name |
| | strFLDS | String | Field list for inclusion (comma-separated) |
| | strTitle | String | Background color |
| | iBGColor | LongInt | Text color |
| | iTextColor | LongInt | Center the table? |
| | lCenter | Logical | Additional HTML table parameters |
| | strParams | String | (filled) Template |
| | mTemplate | Memo | Static output? (FALSE = Dynamic) |
| | lStatic | Logical | Show working messages on status line? |
| | lMsg | Logical | |
| _HTMLReport | strURI | String | URI (or output file name) |
| (record) | strTitle | String | Document title |
| | iBGColor | LongInt | Background color |
| | iTextColor | LongInt | Text color |
| | mHeader | Memo | (filled) Header template |
| | mTemplate | Memo | (filled) Main template |
| | lStatic | Logical | Static output? (FALSE = Dynamic) |
| | lMsg | Logical | Show working messages on status line? |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD_INTRO;',0,"Defaultoverview",)}** **Related Topics**

# HTMLIB01 Corel Paradox HTML Publishing Library methods

The following kinds of ObjectPAL methods are defined in the HTMLIB01 Corel Paradox HTML Publishing Library:

- Publishing methods
- Utility methods
- Repository methods

Only the publishing methods are needed to access the full capabilities of the Corel Paradox HTML Publishing Library. They serve as wrappers for the remaining methods in the library. For an example of how all these methods are used, see the sample applications provided with the Corel Web Server Control. See INDEX.HTM in the main Paradox directory for more information.

---

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD_INTRO;',0,"Defaultoverview",)} Related Topics**

# Publishing methods (Corel Paradox HTML Publishing Library)

Only the publishing methods are needed to access the full capabilities of the Corel Paradox HTML Publishing Library.

- These two methods are complete wrappers for all the functionality in the library:

  HTMLPublish_Table

  HTMLPublish_Report
- You can use these methods to generate HTML templates, precursors for static and dynamic HTML files:

  GenTemplate_Table

  GenTemplate_Report
- You can use this method to convert HTML templates into static and dynamic HTML files:

  GXEngine_Execute

---

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Utility methods (Corel Paradox HTML Publishing Library)

The utility methods are used to extract information from the documents being published and to perform various HTML tasks.

- Use these methods to get the information you need for use with GXEngine_Execute:

  ExtractSourceInfo

  ExtractStaticImages

- Use this method to find and launch a Web browser:

  LaunchBrowser

- Use this method to convert color values into HTML text:

  RGBtoHTML

---

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Repository methods (Corel Paradox HTML Publishing Library)

The repository methods make up the Repository API, the interface to the Web Server Repository where templatized data is stored for dynamic HTML publishing. These methods let developers fully manipulate the contents of the repository. Their functionality is included in the publishing methods.

- Use these methods to manipulate the Web Server Repository tables and their location:
  Repository_Check
  Repository_Create
  Repository_IsIntact
  Repository_GetDir
  Repository_SetDir
- These methods manipulate templates stored in the repository:
  Template_Enum
  Template_GetValue
  Template_EnumEX
  Template_Add
  Template_Remove
- These methods manipulate relationships between templates:
  Relationship_Enum
  Relationship_EnumEX
  Relationship_Add
  Relationship_Remove
- These methods manipulate data sources stored in the repository:
  Source_Enum
  Source_EnumEX
  Source_GetInfo
  Source_Add
  Source_Remove
- These methods manipulate links between data sources:
  Link_Enum
  Link_EnumEX
  Link_GetInfo
  Link_Add
  Link_Remove
- These methods manipulate links between data sources:
  Param_Enum
  Param_EnumEX
  Param_GetValue
  Param_Add
  Param_Remove
- These methods manipulate images stored in the repository:
  Image_Enum
  Image_EnumEX
  Image_GetBinary
  Image_Add
  Image_Remove

---

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# _ExtractSourceInfo (Corel Paradox HTML Publishing Library method)

Extracts source and link information from a Corel Paradox form or report to use with the Corel HTML Engine. To retrieve a list of sources, use **getKeys()** with dynSourceDBs or dynSourceTables.

### Syntax

```
_ExtractSourceInfo(const strDMTable String, var dynSourceDBs _dynString, var dynSourceTables
   _dynString, var dynLinks _dynString, var dynMFLDS _dynString, var dynDFLDS _dynString)
   Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

### Parameters

| | |
|---|---|
| strDMTable | Data Model table to analyze |
| dynSourceDBs | List of source databases (keyed by source):<br>dynSourceDBs["customer"] = ":ALIAS:"<br>dynSourceDBs["orders"] = "C:\\PARADOX\\WORKING"<br>dynSourceDBs["lineitem"] = ":OTHERALIAS:" |
| dynSourceTables | List of source table names (keyed by source):<br>`dynSourceTables["customer"] = "CUSTOMER.DB"`<br>`dynSourceTables["orders"] = "ORDERS.DB"`<br>`dynSourceTables["lineitem"] = "LINEITEM.DB"` |
| dynLinks | List of links (keyed by detail source):<br>dynLinks["orders"] = "customer"<br>dynLinks["lineitem"] = "orders" |
| dynMFLDS | Link fields in master (parent) source:<br>dynMFLDS["orders"] = "Customer No"<br>dynMFLDS["lineitem"] = "Order No" |
| dynDFLDS | Link fields in detail (child) source:<br>dynDFLDS["orders"] = "Customer No"<br>dynDFLDS["lineitem"] = "Order No" |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)}** **Related Topics**

# _ExtractStaticImages (Corel Paradox HTML Publishing Library method)

Converts .BMP graphics to .JPEG format and fills a dynArray (dynImages) with them.

## Syntax

```
_ExtractStaticImages(var rSource Report, var dynImages _dynBinary) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| rSource | The name of the report to search for images |
| dynImages | The DynArray to contain the .JPEG images in binary format |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# _LaunchBrowser (Corel Paradox HTML Publishing Library method)

Searches the client machine for a registered Web browser and launches it to view specified file.

## Syntax

```
LaunchBrowser(strFileName String, lWait Logical) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strFileName | File name to launch in browser |
| lWait | Wait for browser to close? |
| (return) Logical | TRUE with success or FALSE with failure |

**Example**

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

## Code example (_LaunchBrowser method)

```
Uses ObjectPAL
      _LaunchBrowser(strFileName String, lWait Logical) Logical
EndUses
var
      GXEngine       OLEAuto
      Lib            Library
      i              Logical
endVar
i = logical("T") ; declares "i" as being a logical value of T(rue) or F(alse)
      GXEngine.Open ("Corel.GXEngine")
      GXEngine^AddTemplateString ("vendors","<BODY BGCOLOR=#00FF00><BDE_TABLE SRC=\"VENDORS\"
BORDER=10>")
      GXEngine^AddSourceTable ("vendors",fullName(":WORK:"),"vendors.db")
      GXEngine^OutputMethod = 1     ;output to a file
      GXEngine^OutputFileName = "vendors.htm"      ;output to the working directory
      GXEngine^Execute()
      GXEngine^_LaunchBrowser("vendors.htm")
```

# _RGBtoHTML (Corel Paradox HTML Publishing Library method)

Converts an RGB color value to an HTML color string. See the ObjectPAL color constants for a list of colors you can use.

## Syntax

```
_RGBtoHTML(iColor LongInt) String
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| iColor | Color to convert:<br>`iColor = BLUE = 16711680` |
| (return) String | Returns HTML color string:<br>"0000FF" |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# GenTemplate_Table (Corel Paradox HTML Publishing Library method)

Generates an HTML template for a table.

## Syntax

```
GenTemplate_Table(var HTMLTable _HTMLTable) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| HTMLTable | The table to process (defined with parameters included in the _HTMLTable type (see Corel Paradox HTML Publishing Library types) |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# HTMLPublish_Table (Corel Paradox HTML Publishing Library method)

Publishes a table as a static or dynamic HTML document. For a description of the _HTMLTable type, see Corel Paradox HTML Publishing Library types.

## Syntax

```
HTMLPublish_Table(var HTMLTable _HTMLTable) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| HTMLTable | Source table information |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# GenTemplate_Report (Corel Paradox HTML Publishing Library method)

Generates an HTML template for a report.

## Syntax

```
GenTemplate_Report(var rSource Report, var HTMLReport _HTMLReport) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| rSource | The identifier of the report to output as a template |
| HTMLReport | Report record format defined by the _HTMLReport type (see Corel Paradox HTML Publishing Library types) |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# GXEngine_Execute (Corel Paradox HTML Publishing Library method)

Adds templates and parameters to the GXEngine OLE Automation object, then runs the Engine. The GXEngine should already be opened before using this method and initial parameters should be set with the GXEngine Interface OLE Automation methods and properties (output method, path, and file).

You can use _ExtractSourceInfo and _ExtractStaticImages to get most of the parameters needed by GXEngine_Execute.

## Syntax

```
GXEngine_Execute(var GXEngine OLEAuto, strMaster String, var dynTemplates _dynMemo, var
   dynSourceDBs _dynString, var dynSourceTables _dynString, var dynLinks _dynString, var
   dynMFLDS _dynstring, var dynDFLDS _dynString, var dynParams _dynMemo, lMsg Logical) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| GXEngine | GXEngine.DLL, opened as an OLE Automation object |
| strMaster | Data Model table to analyze [xxx true?] |
| dynSourceDBs | List of source databases (keyed by source):<br>dynSourceDBs["customer"] = ":ALIAS:"<br>dynSourceDBs["orders"] = "C:\\PARADOX\\WORKING"<br>dynSourceDBs["lineitem"] = ":OTHERALIAS:" |
| dynSourceTables | List of source table names (keyed by source):<br>`dynSourceTables["customer"] = "CUSTOMER.DB"`<br>`dynSourceTables["orders"] = "ORDERS.DB"`<br>`dynSourceTables["lineitem"] = "LINEITEM.DB"` |
| dynLinks | List of links (keyed by detail source):<br>dynLinks["orders"] = "customer"<br>dynLinks["lineitem"] = "orders" |
| dynMFLDS | Link fields in master (parent) source:<br>dynMFLDS["orders"] = "Customer No"<br>dynMFLDS["lineitem"] = "Order No" |
| dynDFLDS | Link fields in detail (child) source:<br>dynDFLDS["orders"] = "Customer No"<br>dynDFLDS["lineitem"] = "Order No" |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# HTMLPublish_Report (Corel Paradox HTML Publishing Library method)

Publishes a report as a static or dynamic HTML document. For a description of the _HTMLReport type, see Corel Paradox HTML Publishing Library types.

## Syntax

```
HTMLPublish_Report(var rSource Report, var HTMLReport _HTMLReport) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| rSource Report | Report to publish |
| HTMLReport | Report record format defined by the _HTMLReport type (see Corel Paradox HTML Publishing Library types) |
| (return Logical | Returns True with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Image_Enum (Corel Paradox HTML Publishing Library method)

Fills arImages with a list of dynamic image IDs associated with strTemplate.

## Syntax

```
Image_Enum(strTemplate String, var arImages _arLongInt) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strTemplate | The template to search for dynamic images |
| arImages | The list of dynamic image IDs to create |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)}** **Related Topics**

# Image_EnumEX (Corel Paradox HTML Publishing Library method)

Fills dynImages with a list of image IDs and binary values associated with strTemplate. For example, `dynImages["4550"]` = ...binary...

## Syntax

```
Image_Enum(strTemplate String, var dynImages _dynBinary) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strTemplate | The template to search for dynamic images |
| dynImages | The list of dynamic image IDs and associated binary values to create |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Image_GetBinary (Corel Paradox HTML Publishing Library method)

Retrieves an image as binary. You can use Image_Enum or Image_EnumEX to obtain a list of image IDs.

## Syntax

```
Image_GetBinary(iImageID LongInt) Binary
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| iImageID | The ID of the image to retrieve |
| (return) Binary | Returns a binary image with success or blank() with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Image_Add (Corel Paradox HTML Publishing Library method)

Adds a dynamic image and fills in iImageID.

## Syntax

```
Image_Add(strTemplate String, bImage Binary, var iImageID LongInt) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strTemplate | The template to which the image is to be added |
| bImage | The binary image to add |
| iImageID | The ID of the added image (filled in) |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)}** **Related Topics**

# Image_Remove (Corel Paradox HTML Publishing Library method)

Removes an image from a template. You can use Image_Enum or Image_EnumEX to obtain a list of image IDs.

## Syntax

```
Image_Remove(iImageID LongInt) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| iImageID | The ID of the image to remove |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Repository_Check (Corel Paradox HTML Publishing Library method)

Checks for an instance of the Corel Paradox Web Server Repository. If it can’t find one, and there is a server available, it asks the user for a new location.

## Syntax

```
Repository_Check() Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| (return) Logical | TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Repository_Create (Corel Paradox HTML Publishing Library method)

Creates an instance of the Corel Paradox Web Server Repository, used for storing HTML templates.

## Syntax

```
Repository_Create(strDirectory String, lMsg Logical) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strDirectory | Repository directory |
| lMsg | Show working messages? |
| (return) Logical | TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)}** **Related Topics**

# Repository_IsIntact (Corel Paradox HTML Publishing Library method)

Checks a directory for an complete instance of the Corel Paradox Web Server Repository.

## Syntax

```
Repository_IsIntact(strDirectory String) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strDirectory | Repository directory to check |
| (return) Logical | TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Repository_GetDir (Corel Paradox HTML Publishing Library method)

Returns the current Corel Paradox Web Server Repository directory.

## Syntax

```
Repository_GetDir() String
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| (return) String | Returns repository directory, or blank if none is set |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Repository_SetDir (Corel Paradox HTML Publishing Library method)

Sets the directory of the Corel Paradox Web Server Repository.

## Syntax

```
Repository_SetDir(strDirectory String) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strDirectory | New directory for repository |
| (return) Logical | TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Template_Enum (Corel Paradox HTML Publishing Library method)

Enumerates templates in the Corel Paradox Web Server Repository; fills arTemplates with a list of templates of type strType (T, H, R, or blank for all).

## Syntax

```
Template_Enum(strType String, var arTemplates _arString) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strType | Template type (tables, forms, reports)<br>(T, H, R, or blank for all) |
| arTemplates | List of published templates:<br>arTemplates[1] = "CustomerReport"<br>arTemplates[2] = "OrdersTable" |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Template_GetValue (Corel Paradox HTML Publishing Library method)

Retrieves HTML contents of a template in the Corel Paradox Web Server Repository; returns HTML code for a template. You can use Template_Enum to get the name of the template to retrieve.

## Syntax

```
Template_GetValue(strTemplate String) Memo
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strTemplate | Template to retrieve |
| (return) Memo | Returns template HTML contents with success or blank with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Template_EnumEX (Corel Paradox HTML Publishing Library method)

Enumerates templates and respective detail information in the Corel Paradox Web Server Repository; fills dynTemplates with the templates of type strType (T, H, R, or blank for all). For example:
`dynTemplate["customer"] = "<HTML><HEAD... /BODY></HTML>"`

## Syntax

```
Template_EnumEX(strType String, var dynTemplates _dynString) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strType | Template type (tables, forms, reports)<br>(T, H, R, or blank for all) |
| dynTemplates | List of published templates:<br>dynTemplates["CustomerReport"] = "...HTML code..."<br>dynTemplates["OrdersReport"] = "...HTML code..." |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Template_Add (Corel Paradox HTML Publishing Library method)

Adds a template to the Corel Paradox Web Server Repository; adds template contents, mHTML, as a template name strTemplate of type strType.

## Syntax

```
Template_Add(strTemplate String, mHTML Memo, strType String) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strTemplate | Template name to add |
| mHTML | HTML code contents of template |
| strType | Template type (tables, forms, reports)<br>(T, H, or R) |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Template_Remove (Corel Paradox HTML Publishing Library method)

Removes a template from the Corel Paradox Web Server Repository. Optionally cascades to remove all detail (relationships, sources, links, params, inputs, images) information.

## Syntax

```
Template_Remove(strTemplate String,  lCascade Logical) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strTemplate | Template name to remove |
| lCascade Logical | Cascade and remove all detailed information from the repository? (relationships, sources, links, params, inputs, images) |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)}** **Related Topics**

# Source_Enum (Corel Paradox HTML Publishing Library method)

Enumerates the sources for a template in the Corel Paradox Web Server Repository; fills arSources with a list of source names related to strTemplate.

## Syntax

```
Source_Enum(strTemplate String, var arSources _arString) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strTemplate | Template to enumerate |
| arSources | List of data source names:<br>arSources[1] = "customer"<br>arSources[2] = "orders"<br>arSources[3] = "lineitem" |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Source_GetInfo (Corel Paradox HTML Publishing Library method)

Retrieves detailed information about a source in the Corel Paradox Web Server Repository. You can use Source_Enum to get a list of source names in the Web Server Repository.

## Syntax

```
Source_GetInfo(strTemplate String, strSource String, var strDatabase String, var strTable
   String) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strTemplate | Template with source |
| strSource | Data source name to retrieve |
| strDatabase | Data source database |
| strTable | Data source table name |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Source_EnumEX (Corel Paradox HTML Publishing Library method)

Enumerates the sources and respective detail information in the Corel Paradox Web Server Repository:

- Fills dynSourceDBs with the corresponding Database names
- Fills dynSourceTables with the corresponding Table names

You can use **getKeys** to retrieve a list of source names.

### Syntax

```
Source_EnumEX(strTemplate String, var dynSourceDBs _dynString, var dynSourceTables _dynString)
   Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

### Parameters

| | |
|---|---|
| strTemplate | Template to enumerate |
| dynSourceDBs | List of source Databases (keyed by source):<br>dynSourceDBs["customer"] = ":ALIAS:"<br>dynSourceDBs["orders"] = "C:\\PARADOX\\WORKING"<br>dynSourceDBs["lineitem"] = ":OTHERALIAS:" |
| dynSourceTables | List of source table names (keyed by source):<br>dynSourceTables["customer"] = "CUSTOMER.DB"<br>dynSourceTables["orders"] = "ORDERS.DB"<br>dynSourceTables["lineitem"] = "LINEITEM.DB" |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Source_Add (Corel Paradox HTML Publishing Library method)

Adds a source to the Corel Paradox Web Server Repository.

## Syntax

```
Source_Add(strTemplate String, strSource String, strDatabase String, strTable String) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strTemplate | Template name to which to add new source |
| strSource | Data source name to add |
| strDatabase | Data source database |
| strTable | Data source table name |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Source_Remove (Corel Paradox HTML Publishing Library method)

Removes a source from the Corel Paradox Web Server Repository. Optionally cascades to remove all detail (link) information.

## Syntax

```
Source_Remove(strTemplate String, strSource String, lCascade Logical) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strTemplate | Template name from which to remove source |
| strSource | Data source name to remove |
| lCascade | Cascade and remove all detailed information from the Web Server Repository? (links) |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Link_Enum (Corel Paradox HTML Publishing Library method)

Enumerates the links for a template in the Corel Paradox Web Server Repository; fills dynLinks with a list of links, keyed by detail source name.

## Syntax

```
Link_Enum(strTemplate String, var dynLinks _dynString) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strTemplate | Template to enumerate |
| dynLinks | List of links (keyed by detail table):<br>dynLinks["orders"] = "customer"<br>dynLinks["lineitem"] = "orders" |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)}** **Related Topics**

# Link_GetInfo (Corel Paradox HTML Publishing Library method)

Retrieves detailed information about a link in the Corel Paradox Web Server Repository.

## Syntax

```
Link_GetInfo(strTemplate String, strDSource String, var strDFLDS String, var strMSource String,
   var strMFLDS String) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strTemplate | Template with links |
| strDSource | Detail (child) data source name |
| strDFLDS | Link fields in detail (child) table |
| strMSource | Master (parent) data source name |
| strMFLDS | Link fields in master (parent) table |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Link_EnumEX (Corel Paradox HTML Publishing Library method)

Enumerates the links and respective detail information for a template in the Corel Paradox Web Server Repository

- fills dynLinks with a list of links (keyed by detail source name) used by strTemplate
- fills dynDFLDS with the corresponding fields from the child table for each link
- fills dynMFLDS with the corresponding fields from the parent table for each link

### Syntax

```
Link_EnumEX(strTemplate String, var dynLinks _dynString, var dynDFLDS _dynString, var dynMFLDS
   _dynString) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

### Parameters

| | |
|---|---|
| strTemplate | Template with links to enumerate |
| dynLinks | List of links (keyed by detail table):<br>dynLinks["orders"] = "customer"<br>dynLinks["lineitem"] = "orders" |
| dynDFLDS | Link fields in detail (child) table:<br>dynDFLDS["orders"] = "Customer No"<br>dynDFLDS["lineitem"] = "Order No" |
| dynMFLDS | Link fields in master (parent) table:<br>dynMFLDS["orders"] = "Customer No"<br>dynMFLDS["lineitem"] = "Order No" |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Link_Add (Corel Paradox HTML Publishing Library method)

Adds a link between sources to the Corel Paradox Web Server Repository.

## Syntax

```
Link_Add(strTemplate String, strDSource String, strDFLDS String, strMSource String, String)
   Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strTemplate | Template containing sources |
| strDSource | Detail (child) data source name |
| strDFLDS | Link fields in detail (child) table |
| strMSource | Master (parent) data source name |
| strMFLDS | Link fields in master (parent) table |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Link_Remove (Corel Paradox HTML Publishing Library method)

Removes a link between sources from the Corel Paradox Web Server Repository.

## Syntax

```
Link_Remove(strTemplate String, strDSource String) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strTemplate | Template name from which to remove link |
| strDSource | Detail (child) data source name |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Relationship_Enum (Corel Paradox HTML Publishing Library method)

Enumerates related templates to a template in the Corel Paradox Web Server Repository; fills arTemplates with a list of template names related to strTemplate.

## Syntax

```
Relationship_Enum(strTemplate String, var arTemplates _arString) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strTemplate | Template to enumerate relationships |
| arTemplates | List of related templates:<br>arTemplates[1] = "CustomerReport"<br>arTemplates[2] = "OrdersTable" |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Relationship_EnumEX (Corel Paradox HTML Publishing Library method)

Enumerates related templates to a template in the Corel Paradox Web Server Repository; fills dynTemplates with a list of template names related to strTemplate (keyed by template name). Value = template type (h header, F footer, other = HTML).

## Syntax

```
Relationship_EnumEX(strTemplate String, var dynTemplates _dynString) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strTemplate | Template to enumerate relationships |
| dynTemplates | Keyed list of related templates |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Relationship_Add (Corel Paradox HTML Publishing Library method)

Adds a template relationship to the Corel Paradox Web Server Repository.

## Syntax

```
Relationship_Add(strMTemplate String, strDTemplate String, strType String) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strMTemplate | Master template |
| strDTemplate | Detail (related) template |
| strType | H = header, F = footer, anything else for raw HTML |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)}** **Related Topics**

# Relationship_Remove (Corel Paradox HTML Publishing Library method)

Removes a template relationship from the Corel Paradox Web Server Repository.

## Syntax

```
Relationship_Remove(strMTemplate String, strDTemplate String) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strMTemplate | Master template |
| strDTemplate | Detail (related) template |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)}** **Related Topics**

# Param_Enum (Corel Paradox HTML Publishing Library method)

Enumerates params for a template in the Corel Paradox Web Server Repository; fills arParams with a list of PARAM names used in strTemplate.

## Syntax

```
Param_Enum(strTemplate String, var arParams _arString) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strTemplate | Template to enumerate |
| arParams | List of params:<br>arParams[1] = "TIMESTAMP"<br>arParams[2] = "WelcomeText" |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)}** **Related Topics**

# Param_GetValue (Corel Paradox HTML Publishing Library method)

Retrieves a PARAM value from the Corel Paradox Web Server Repository.

## Syntax

```
Param_GetValue(strTemplate String, strParam String) Memo
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strTemplate | Template containing PARAM |
| strParam | PARAM to retrieve |
| (return) Memo | Returns param value with success or blank with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Param_EnumEX (Corel Paradox HTML Publishing Library method)

Enumerates params and respective detail information for a template in the Corel Paradox Web Server Repository; fills synParams with a list of PARAM names and values used in strTemplate. For example:
`dynParams["RULER"] = "<HR>"`

## Syntax

```
Param_EnumEX(strTemplate String, var dynParams _dynMemo) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strTemplate | Template to enumerate |
| dynParams | List of PARAM values (keyed by PARAM):<br>arParams["TIMESTAMP"] = "June 10, 1996"<br>arParams["WelcomeText"] = "Hello, how are you?" |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Param_Add (Corel Paradox HTML Publishing Library method)

Adds a PARAM to the Corel Paradox Web Server Repository.

## Syntax

```
Param_Add(strTemplate String, strParam String, mValue Memo) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strTemplate | Template to which to add param |
| strParam | PARAM name |
| mValue | PARAM value |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)} Related Topics**

# Param_Remove (Corel Paradox HTML Publishing Library method)

Removes a PARAM from the Corel Paradox Web Server Repository.

## Syntax

```
Param_Remove(strTemplate String, strParam String) Logical
```

**Note**

- The above syntax should be copied without modification for the method to function properly.

## Parameters

| | |
|---|---|
| strTemplate | Template with PARAM to remove |
| strParam | PARAM to remove |
| (return) Logical | Returns TRUE with success or FALSE with failure |

**{button ,AL(`HTMLIBSPEC;HTMLIBMETHOD;',0,"Defaultoverview",)}** **Related Topics**

# Hyperlink capabilities in Corel Paradox 8

Corel Paradox 8 now supports hyperlinks to the Internet as static design objects and as data in form and table fields.

A text object can be formatted as a hyperlink when created using the Text Expert. It is also possible to format an existing text object as a hyperlink by adding ObjectPAL code to the mouseClick event in the Object Explorer.

You can insert a hyperlink as data by simply typing the URL in a form or table field.

When you click on a hyperlink, Corel Paradox will use the appropriate Internet protocol to follow the link (e.g., for HTTP it will launch the default browser to display the website, for MAILTO it will open a new message in the default mail program, etc.).

---

**{button ,AL(`HYPERLINK;',0,"Defaultoverview",)} Related Topics**

## Inserting hyperlinks

Hyperlinks to the Internet can be inserted as data in alpha fields in tables and forms, or as design objects on forms and reports.

**To insert a hyperlink in a field**

- Type the URL in an alpha field.

**To insert a hyperlink as a design object**

1. Open a form in design mode.
2. Using the text tool, place a new text object on the form. Corel Paradox launches the Text Expert.
3. Follow the steps in the Text Expert. In the last step, enable the Hyperlink button.
4. Type the URL in the Hyperlink box and click Finish.
5. Run the form to activate the hyperlink.

**Notes**

- Be sure to include the appropriate protocol for the hyperlink (e.g., HTTP://, FTP://).
- You do not have to use the actual URL as the text to be linked. Whatever text you use will be linked to the URL specified.

---

**{button ,AL(`HYPERLINK;',0,"Defaultoverview",)} Related Topics**

## Supported protocols for hyperlinks

Corel Paradox supports the following Internet protocols for hyperlinks:

- HTTP
- FTP
- MAILTO
- GOPHER
- NEWS
- TELNET

---

**{button ,AL(`HYPERLINK;',0,"Defaultoverview",)}** **Related Topics**

# About HTML Publishing in Corel Paradox 8

The following Corel Paradox 8 features make publishing to HTML easier than ever:

- the HTML Report Expert
- the HTML Table Expert
- the ability to save Corel Paradox forms as HTML files
- HyperText template (.HTT) files

---

**{button ,AL(`PUBLISH;',0,"Defaultoverview",)}** **Related Topics**

# About the Corel Paradox HTML Publishing Experts

The Corel Paradox HTML Publishing Experts help you change your Paradox files into HTML files.

### HTML Table Expert

The HTML Table Expert converts a table to an HTML text file so you can publish the table on the World Wide Web. The Expert puts in the appropriate HTML tags and parameters automatically. You can modify the HTML document created by the Expert, as you would any other HTML document. Click File, Publish to HTML to launch the HTML Table Expert.

### HTML Report Expert

The HTML Report Expert helps you convert a report to an HTML text file so you can publish the report on the World Wide Web. The Expert puts in the appropriate HTML tags and parameters automatically. You can later modify the resulting HTML document, as you would any other HTML document. Click File, Publish to HTML to launch the HTML Report Expert.

---

**{button ,AL(`PUBLISH;',0,"Defaultoverview",)} Related Topics**

# About the HTML Import Expert

The HTML Import Expert helps you import tables or lists from HTML files into Corel Paradox tables. It will let you add the data to an existing table or help you create a new one for your data. Click File, Import to launch the HTML Import Expert.

For help using the HTML Import Expert, click the Help button in the Expert’s dialog box.

---

**{button ,AL(`IMPORT;',0,"Defaultoverview",)}** **Related Topics**

## Importing data from an HTML file

**To import data from an HTML file**

1. Click File, Import.
2. Click HTML Import Expert button in the Import dialog box.
3. Follow the steps in the HTML Import Expert.

---

**{button ,AL(`IMPORT;',0,"Defaultoverview",)}** **Related Topics**

# About .HTT files

An .HTT (HyperText Template) file is an HTML file with custom metatags. These tags are Paradox-specific, and are used by the HTML Publishing Engine to produce standard HTML files. HTT files can be viewed as HTML in a browser as is, thought the browser will ignore all occurences of custom tags. Conceptually, an .HTT file is an intermediary step in the Internet publishing system; all layout information is defined but metatags are used in place of tables, queries, and other database objects.

HTT files are generally used in dynamic publishing where the layout information is known but the actual data needs to be resolved at run time. The dynamic publishing of the Corel Web Server (.EXE) manages .HTT files automatically, so their existence and function is largely transparent to the user. If you are implementing your own dynamic publishing system with the Corel Web Server Control, you can generate .HTT files using File, Publish to HTML for tables, reports, and forms.

---

**{button ,AL(`PUBLISH;',0,"Defaultoverview",)}** **Related Topics**

# About publishing Corel Paradox forms to HTML

In addition to the HTML Report Expert and the HTML Table Expert, Corel Paradox can also publish any Corel Paradox form to a static HTML document. This feature is useful for creating an exact replica of your Paradox form for use on the Internet, eliminating the need to recreate it in an HTML editor.

---

**{button ,AL(`PUBLISH;',0,"Defaultoverview",)}** **Related Topics**

# About the Web Server Repository

Use the Web Server Repository to store, view, and edit GXEngine templates for dynamic HTML publishing. It is a storage facility for HTML template files that are created by the Corel Paradox HTML Publishing Experts and used by the Corel Web Server Control and the GXEngine to produce dynamic HTML files on browser requests.

Click Tools, Web Server Repository to view a list of stored templates and their contents.

**Note**

- You can drag a table or report from the Project Viewer to the Repository page of the Web Server Repository to launch the appropriate HTML Publishing Expert.

---

**{button ,AL(`OVERVIEW;',0,"Defaultoverview",)}** **Related Topics**

# About the Corel Web Servers

The Corel Paradox Internet tools include two Web servers which support HTTP 1.0 (and some HTTP 1.1 protocols). You can use the Corel Web Server (.EXE) and the Corel Web Server Control to process client browser requests and send responses from compatible applications such as Corel Paradox. The Web servers also handle special requests which are unique to compatible applications.

### How do they differ?

The Corel Web Server (.EXE) is a complete application which easily publishes static or dynamic reports or tables from Paradox. It cannot, however, get user information to send back custom responses and it does not have the capability to process or respond to posted electronic forms.

The Corel Web Server Control is an ActiveX control which is placed on a Corel Paradox form as an OLEobject. Like other Corel Paradox objects, it can be programmed using its own set of ObjectPAL properties, methods and events to send custom responses to client requests. It is also able to process electronic forms and write information to Corel Paradox tables.

---

**{button ,AL(`WEBSRV;',0,"Defaultoverview",)} Related Topics**

# About the Corel Web Server (.EXE)

The Corel Web Server executable file (.EXE) is a fully functional Web server with the following features:

- HTTP support (version 1.0 and many 1.1 elements)
- file caching
- contact (access) logging using the CERN/NCSA Common Log Format plus transaction logging
- custom MIME types
- multi-threaded (supports multiple IntraBuilder sessions)

The Corel Web Server acts as an intermediary between Web browsers and compatible applications, such as IntraBuilder and Corel Paradox, to transfer requests and responses between them.

For more information, see the Corel Web Server (.EXE) Help file.

---

**{button ,AL(`OVERVIEW;',0,"Defaultoverview",)} Related Topics**

# About the Corel Web Server Control

The Corel Web Server ActiveX control is a fully functioning Internet server with these features:

- HTTP 1.0 support (plus some features of HTTP 1.1)
- file caching
- logging using the CERN/NCSA Common Log Format
- custom MIME types
- support for cookies (sessions)
- support for multi-threaded containers, such as Delphi
- OGI programming interface

To use the Corel Web Server Control, place the ActiveX control in any product that supports OLE controls as a container, for example, a Corel Paradox form.

The container is notified when an HTTP GET, POST, or HEAD method is requested by a Web browser client. The Corel Web Server Control supplies its container with an OGI (OLE Gateway Interface) event. An OGI event is an ActiveX control ConnectionPoint event. This allows the container to execute an event-handling procedure in its native code. Custom event behavior can be programmed or the default HTTP response can be accepted.

---

**{button ,AL(`OVERVIEW;',0,"Defaultoverview",)} Related Topics**

# About MIME types

Multi-purpose Internet Mail Extensions (MIME) developed as a way to send text formatting, graphics, sounds, and other multimedia elements across different Internet mail systems. They are now also used as a way to help Web browsers and servers communicate. For example, a browser will include a list of MIME types it supports when it sends a request to a server. The server responds by attempting to return the requested data in a format the browser can understand.

MIME establishes relationships between files and their content by mapping file extensions to specific MIME types, as shown by the examples in the table below.

| File extension | MIME type |
|---|---|
| .HTM, .HTML | text/html |
| .TXT | text/plain |
| .JPG, .JPEG | image/jpeg |
| .GIF | image/gif |
| .WAV | audio/wav |
| .MPG, .MPEG | video/mpeg |

You can modify and delete MIME types from the Corel Web Server MIME types database. The Corel Web Server also supports custom MIME types.

---

**{button ,AL(`MIME;',0,"Defaultoverview",)}** **Related Topics**

## Adding custom MIME types

**To add a custom MIME type to the Corel Web Server Control MIME type database**

1. Right-click the Corel Web Server Control object and click Properties, Corel Web Server Control.
2. Click the MIME tab.
3. Click the Add button.
4. Type the file extension, the corresponding MIME type, and the subtype.

Your new MIME type now appears in the Corel Web Server Control MIME types database.

---

**{button ,AL(`MIME;',0,"Defaultoverview",)}** **Related Topics**

## Deleting MIME types

**To delete a MIME type from the Corel Web Server Control MIME type database**

1. Right-click the Corel Web Server Control object and click Properties, Corel Web Server Control.
2. Click the MIME tab.
3. From the list box, select the MIME type you want to remove.
4. Click the Remove button.

---

**{button ,AL(`MIME;',0,"Defaultoverview",)}** **Related Topics**

## Modifying MIME types

**To modify an existing MIME type**

1. Right-click the Corel Web Server Control object and choose Properties, Corel Web Server Control.
2. Click the MIME tab.
3. From the list box, select the MIME type you want to modify.
4. Click the Edit button.
5. Make your changes to the MIME type or subtype.

Your MIME type is now modified in the Corel Web Server Control MIME types database.

---

**{button ,AL(`MIME;',0,"Defaultoverview",)}** **Related Topics**

# Running multiple instances of the Corel Web Server Control

Using several Corel Web Server Control objects (on the same form or on different forms) to handle different sets of pages can reduce the load on a single server and may result in shorter response time. However, if you want to have multiple active Web servers, each must be attached to a different port. Since most browser requests come in at the default port 80, servers waiting for requests at a different port may remain idle.

**Note**

- If you have multiple open forms containing a Corel Web Server Control object set to port 80, Corel Paradox uses the form you opened first as the default Web server.

---

**{button ,AL(`USING;',0,"Defaultoverview",)}** **Related Topics**

# Interoperability with the Corel Web Server (.EXE)

The Corel Web Server Control will override the Corel Web Server (.EXE). If an open form in Corel Paradox contains a Corel Web Server Control object, it is the server for all incoming browser requests.

**Note**

- If Corel Paradox is running but there are no open forms containing a Corel Web Server Control object, the Corel Web Server (.EXE) will handle incoming browser requests if it is active.

---

**{button ,AL(`USING;',0,"Defaultoverview",)}** **Related Topics**

## Code Example 1

The following example uses these methods and properties:

- Request.URI
- Request.GetField()
- Response.ResultString
- Response.StatusCode

### Example

```
;// This example insert records into the contact table.
;//
method OnPostRequest(Request OleAuto, Response OleAuto)
var
  tc TCursor
  name  String
endVar

if (Request.URI = "/ADDCONTACTINFO") then
     try
        tc.open(":sample:Contacts.DB")
        tc.edit()
        tc.insertRecord()

        ;// Use GetField() to get the value from each
        ;// field on the HTML form by the field name.

        name = Request.GetField("FirstName")
        tc."First Name" = name
        tc."Last Name" = Request.GetField("LastName")
        tc."Company" = Request.GetField("Company")
        tc."Phone" = Request.GetField("Phone")

        tc.unlockRecord()
        tc.close()

        ;// Set the ResultString to send content back
        ;// to the user's browser.

        Response.ResultString = string(
           "<HTML><H1>Thank You ", name, "!</H1><BR>",
           "We will be contacting you soon.</HTML>" )

     onFail

        ;// Set the Status to 500 Server Error if
        ;// this try block fails.

        Response.StatusCode = 500
        Response.ResultString = string(
           "<HTML><H1>SERVER ERROR!</H1><BR>",
           "Could not add your information.<BR>",
           ErrorMessage(), "</HTML>" )

     endTry
  endIf

endMethod
```

## Code Example 2

The following example uses these methods and properties:

- Request.NFields
- Request.GetFieldName()
- Request.GetFieldByIndex()

### Example

```
method OnPostRequest(Request OleAuto, Response OleAuto)
var
  dyn   DynArray[] String
  n     SmallInt
endVar

  ;// Enumerate all the HTML fields and show
  ;// them in a DynArray.

  n = Request.NFields - 1

  ;// GetFieldName and GetFieldByIndex use zero based
  ;// indexes, so start with 0 and end with NField - 1.

  for i from 0 to n

     dyn[ Request.GetFieldName(i) ] = Request.GetFieldByIndex(i)

  endFor

  dyn.view()

endMethod
```

## Code Example 3

The following example uses these methods and properties:

- Request.URI
- Request.UserAgent
- Response.ResultFile

### Example

```
method OnGetRequest(Request OleAuto, Response OleAuto)
var
  browser  String
endvar

  ;// Use the UserAgent property to find which browser
  ;// the user is using.  If they are using Netscape
  ;// then send a default page with frames.

  if Request.URI = "/" then

     browser = Request.UserAgent

     ;// Netscape = Mozilla
     if browser.search("Mozilla") > 0 then

        Response.ResultFile = "c:\\webpages\\indexwithframes.html"

     endIf

  endIf

endMethod
```

## Code Example 4

The following example uses these properties:

- Request.IPAddress
- Response.ResultFile

### Example

```
;//  The example filters the request based on the IP address
;//
method OnGetRequest(Request OleAuto, Response OleAuto)
var
  ip String
endVar

  ip = Request.IPAddress

  ;// Check if IP Address came from Corel.
  ;// If not send the external page.

  if NOT ip.advMatch("^143.186.*") then

     Response.ResultFile = "c:\\webpages\\external.html"

  endIf


endMethod
```

## Code Example 5

The following example uses these properties:

- Request.IPAddress
- Request.ClientName

### Example

```
;//  The example filters the request based on the ClientName
;//
method OnGetRequest(Request OleAuto, Response OleAuto)
var
  hostname String
endVar

  hostname = Request.ClientName

  ;// Check if Request came from Corel.
  ;// If not send the external page.

  ignoreCaseInStringCompares ( YES )

  if hostname.search("Corel.com") = 0 then

     Response.ResultFile = "c:\\webpages\\external.html"

  endIf

endMethod
```

## Code Example 6

The following example uses these properties:

- Request.AuthorizationUserid
- Request.AuthorizationPassword
- Response.StatusCode
- Response.WWWAuthenticateRealm
- Response.ResultString

### Example

```
;//  This example shows user authentication.
;//
method OnGetRequest(Request OleAuto, Response OleAuto)
var
  user, password             String
endVar

  user = Request.AuthorizationUserid
  password = Request.AuthorizationPassword
  password.view()
  user.view()

  if user <> "Wally" OR password <> "Corndog" then

     ;// Not Authorized
     Response.StatusCode = 401
     Response.WWWAuthenticateRealm = "Wally World"
     Response.ResultString = "<HTML>You are not authorized.
     Please enter your UserName and Password. </HTML>"

  endIf

endMethod
```

## Code Example 7

The following example uses these properties:

- Request.Date
- Response.ResultString

### Example

```
;//  Display the date sent by the browser
;//
method OnGetRequest(Request OleAuto, Response OleAuto)
var
  dt DateTime
endVar

  dt = Request.Date

  Response.ResultString = "<HTML>  Your request made on :" + String(dt) + "</HTML>"

endMethod
```

## Code Example 8

The following example uses these properties:

- Request.URI
- Request.Accept
- Response.ResultFile
- Response.ContentType

### Example

```
;// This example switches a GIF image request to a JPG
;// image request if the user can accept JPGs.
method OnGetRequest(Request OleAuto, Response OleAuto)
var
  accept   String
endVar

  if Request.URI = "/globe.gif" then

     accept = Request.Accept

     ;// Search the Accept Field to if they can accept JPG images.

     if accept.search("image/jpg") > 0 then

        ;// Send the JPG version of the image

        Response.ResultFile = "c:\\webpages\\globe.jpg"

        ;// Change the content type to JPG

        Response.ContentType = "image/jpg"

     endIf

  endIf

endMethod
```

## Code Example 9

The following example uses these properties:

- Request.URI
- Request.QueryString
- Response.ResultString

### Example

```
;// This example uses a QueryString to search for orders
;// in the Order table of the Samples directory.
;// An example using a QueryString would be to type the
;// following after your base path entry into your browser:
;// GETCUSTORDERS?customerno=1221

method OnGetQueryRequest(Request OleAuto, Response OleAuto)

var
  qbe      Query
  tc    TCursor
  result   String
  toks     Array[] String
  qstr     String
endVar

  if (Request.URI <> "/GETCUSTORDERS") then return endIf

  ;// Split up the Name=Value pair
  qstr = Request.QueryString
  qstr.breakApart(toks,"=")

  ;// Build & Run Query
  qbe.appendTable(":Sample:Orders.DB")
  qbe.checkRow(":Sample:Orders.DB",CheckCheck)
  qbe.setCriteria(":Sample:Orders.DB","Customer No",toks[2])

  qbe.executeQBE(tc)

;// Convert Query output to HTML

  ;// Create HTML Table Header
  result = string(
     "<HTML><P><TABLE BORDER=2>",
     "<TR><TH>Order No</TH>",
     "<TH>Total Invoice</TH>")

  ;// Fill in HTML Table Contents
  scan tc : result = result + string(
     "<TR>",
     "<TD>", tc."Order No", "</TD>",
     "<TD>", tc."Total Invoice" ,"</TD>")
  endScan

  Response.ResultString = result + "</TABLE></HTML>\r\n"

endMethod
```

## Code Example 10

The following example uses these properties:

- Request.URI
- Request.GetCookie
- Response.SetCookie
- Response.ResultString

### Example

```
method OnGetRequest(Request OleAuto, Response OleAuto)

var
  color, newcolor String
endVar

  if Request.URI = "/togglecolors.htm" then

     color = Request.GetCookie("Color")

     if color.isBlank() then
        color = "Black"
     endIf

     switch
        case color = "Purple":
           ;// Purple -> Black
           newcolor = "<BODY BGCOLOR=#330000 TEXT=#FF6666>"
           Response.SetCookie("Color","Black",Date("1/1/99"),"","/")

        case color = "Black":
           ;// Black -> Green
           newcolor = "<BODY BGCOLOR=#00FF33 TEXT=#0000CC>"
           Response.SetCookie("Color","Green",Date("1/1/99"),"","/")

        case color = "Green":
           ;// Green -> Purple
           newcolor = "<BODY BGCOLOR=#CC99FF TEXT=#000000>"
           Response.SetCookie("Color","Purple",Date("1/1/99"),"","/")

     endSwitch

     Response.ResultString = "<HTML>" + newcolor + "Reload the page to watch the colors change.
</HTML>"

  endIf

endMethod
```

# About OGI

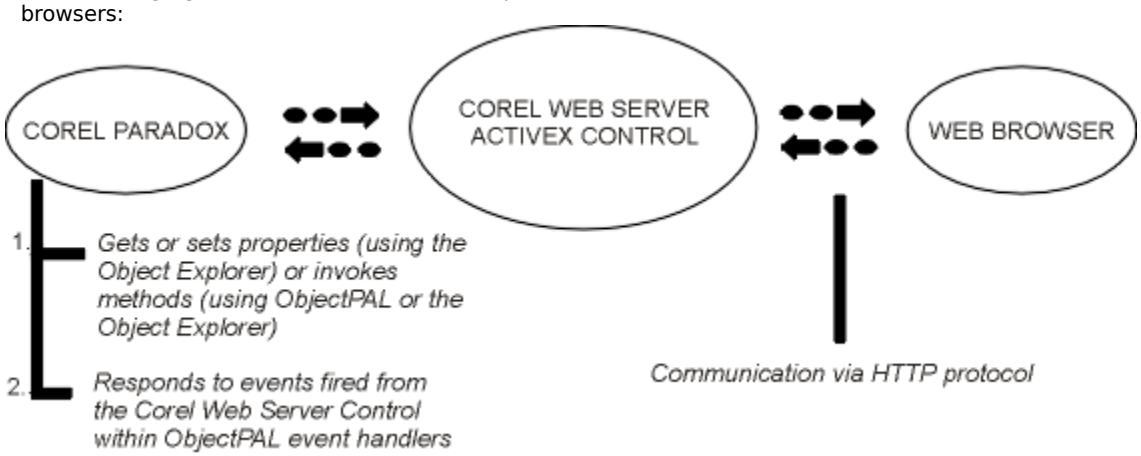
The internal Corel Web Server Control manager fires a different event for each type of HTTP method it receives from a browser: GET, POST, or HEAD. The parameters passed for these events are Request and Response, type OLEAuto, which are pointers to OLE IDispatch objects. Users can get or put properties and call methods on the Request and Response to send and get information from the server. This is known as OGI or OLE Gateway Interface.

As currently implemented, an OGI event is an ActiveX control ConnectionPoint event. This allows the container to execute an event-handling procedure in its own native code (ObjectPAL). Custom event behavior can be programmed or the default HTTP response can be accepted.

Much like CGI, you can use OGI to connect the server and container (database processor, in this case Corel Paradox). But the OGI interface follows the HTTP specification much more closely than CGI, giving more information and control over the server. For example, a common event handler would read the URI and ClientName properties from Request and then set the ResultString property of Response to send back a dynamic Web page.

For demonstration forms with ObjectPAL code examples, see Corel Web Server Control sample applets.

The following figure illustrates the relationship between Corel Paradox, the Corel Web Server Control, and Web browsers:




### HTTP specification

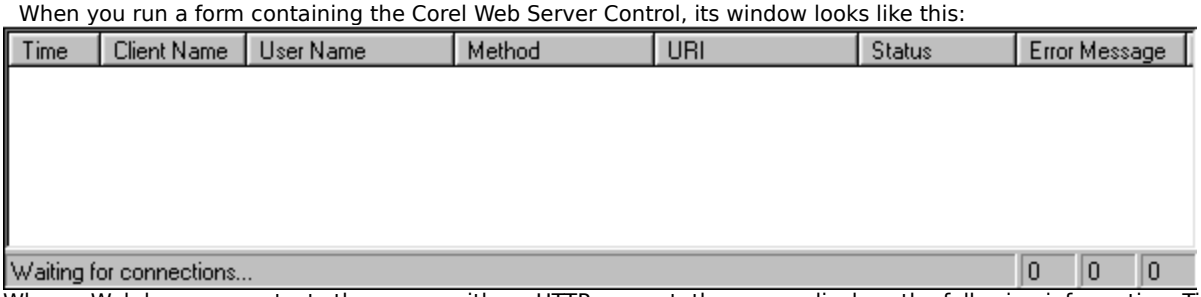
To access various drafts of the Hypertext Transfer Protocol (HTTP) specifications, browse the following URL on the World Wide Web: http://www.ics.uci.edu/pub/ietf/http.

---

**{button ,AL(`ABOUT;',0,"Defaultoverview",)}** **Related Topics**

# The Corel Web Server Control user interface

When you run a form containing the Corel Web Server Control, its window looks like this:




When a Web browser contacts the server with an HTTP request, the server displays the following information: Time = contact time, Client Name = Request object ClientName property value, User Name = Request object AuthorizationUserid property value, Method = HTTP method used in the request: GET, POST, or HEAD, URI = Request object URI property value, Status = the HTTP connection code for the current status of the contact, Error Message = an error message. Dead connections show only if ShowCompletedConnections is set to True on the Corel Web Server Control properties Server page.

If UpdateServerStatusBar is set to True (the default), status messages appear on the Status Bar at the bottom of the server. The first number at the right end of the Status Bar shows the number of current connections, the next number shows the total number of connections, and the third number shows the number of failed connections.

---

**{button ,AL(`USING;',0,"Defaultoverview",)} Related Topics**

# Customizing the Corel Web Server Control

You can customize the Corel Web Server Control by setting its properties and attaching ObjectPAL code to its events and methods.

### Setting Corel Web Server Control properties

To set the Corel Web Server Control's properties, right-click the Corel Web Server Control in the Form Design window and click Properties, Corel Web Server Control.

For more information, see About Corel Web Server Control properties.

One important property to set is the Base Path, on the Corel Web Server Control properties Pages page. This property sets the root directory where target Web pages are stored (which should not be the computer's root directory).

### Attaching code to the Corel Web Server Control

You can program the behavior of the Corel Web Server Control by attaching ObjectPAL code to its events.

For example, to read HTML pages from a database (or create them on the fly), you can add code to the OnGetRequest event to locate the page in the table and add it to the response record (about six to eight lines of code, as opposed to an entire program with conventional CGI). Likewise, to process forms for example, to post records and run queries just add code to the OnPostRequest event.

---

**{button ,AL(`USING;',0,"Defaultoverview",)} Related Topics**

## Processing information from HTML input forms

You can configure the Corel Web Server Control to trap information submitted by HTML input forms in its OnPostRequest event. This example uses an HTML input form named SIMPLEFORM.HTM with the following characteristics:

```
<HTML>
<HEAD>
<TITLE>Simple Input Form</TITLE>
</HEAD>

<H1 ALIGN=CENTER><B>Enter Your Name</B></H1>
<P><HR></P>

<P><FORM ACTION="FormData1" METHOD="POST">
<B>Name:</B><BR>
<INPUT NAME="Name"></P>

<P><INPUT TYPE="SUBMIT" VALUE="Submit"></FORM></P>

</BODY>
</HTML>
```

You can cut and paste the above text into an HTML file to try the following example.

**To trap information posted by an HTML input form**

1. Design the Corel Paradox form containing the Corel Web Server Control object.
2. Set the Base Path of the Corel Web Server Control to the form SIMPLEFORM.HTM on the Pages page in the Corel Web Server Control Properties dialog box.
3. Right-click the Corel Web Server Control object and click Object Explorer.
4. Click the Event tab.
5. Double-click the OnPostRequest event to open the Editor window.
6. Add the following code:

```
method OnPostRequest(Request OleAuto, Response OleAuto)
var
     vname string
endvar
  if request.URI="/FormData1" then
     vname=request.getfield("Name")

           Response.ResultString = string(
     "<HTML><H1>Thank You ", vname, "!</H1></HTML>" )
  endif
endMethod
```

You can request SIMPLEFORM.HTM from the Corel Web Server Control using your browser. When you click the Submit button, the Corel Web Server Control is prepared to process the information you have typed in the form.

**Note**

- Create HTML input forms by using File, Publish to HTML in Corel Paradox or using an HTML editor such as Corel Web Designer.

# Corel Web Server Control sample applets

The Corel Web Server Control includes two small sample applets and a larger application to demonstrate its capabilities.

Sample 1: INSERT.FSL a simple posting example

Sample 2: QUERY.FSL a simple querying example

Sample 3: SERVER.FSL a corporate Web site example that uses tcursors and scan loops to create tables on the fly. It also includes extensive use of HTML "snippets" to create templates for parts of pages.

---

**{button ,AL(`APPLETS;',0,"Defaultoverview",)}** **Related Topics**

## Sample 1: INSERT.FSL

This application takes your name, company, and phone number, then posts it to a table when you click a button. Following that, a confirmation message appears to thank you for your entry.

**To run the application**

1. Set the Corel Paradox working directory to WEBSRV\SAMPLE under the Paradox program directory.
2. Double-click INSERT.FSL to run the form containing the Corel Web Server Control.
3. Start your Web browser and specify your machine name or IP address as the location to open.

   The default page is set in the Corel Web Server Control Properties so, when you open your machine location with the form running, the server knows the correct page to send to the browser. (To view the Web Server Control Properties dialog box, right click the Web Server Control in Design mode and click Properties, Corel Web Server Control.)
4. Type your name, company, and telephone number in the appropriate boxes, then click the Post Contact button. You will receive a confirmation message.

**Notes**

- If you maximize Corel Paradox now, you will notice that the POST and GET requests generated by the browser are displayed in the Corel Web Server Control on the INSERT.FSL form.
- To see the underlying code, click Form, Design Form when viewing the server, then right-click the server and click Object Explorer. Double-click the OnPostRequest event to look at the code. You can see how the field contents are obtained and the response is generated.
- The information gathered is written to CONTACTS.DB in the WEBSRV\SAMPLE directory. If you open this table, you will see that the information you just submitted has been added to the existing contact information.
- To see the HTML code used in this application, open INSERT.HTM in WEBSRV\SAMPLE\PAGES.

---

**{button ,AL(`APPLETS;',0,"Defaultoverview",)} Related Topics**

## Sample 2: QUERY.FSL

This application takes a customer number from ORDERS.DB and returns associated orders.

**To run the application**

1. Set the Corel Paradox working directory to WEBSRV\SAMPLE under the Paradox program directory.
2. Double-click QUERY.FSL to run the form containing the Corel Web Server Control.
3. Start your Web browser and specify your machine name or IP address as the location to open.

   The default page is set in the Corel Web Server Control Properties so, when you open your machine location with the form running, the server knows the correct page to send to the browser. (To view the Web Server Control Properties dialog box, right-click the Web Server Control in Design mode and click Properties, Corel Web Server Control.)
4. Choose a query option:
   - Simple Query allows you to view the order numbers for a specific customer by entering the appropriate customer number
   - Query with Dynamic List Box demonstrates Corel Paradox's ability to generate a list box from data in a table
5. Specify the customer number (for the Simple Query option) or choose a customer name from the drop-down list box (for the Query with Dynamic List Box option).
6. Click the Show Orders button.

A table showing the orders for the specified customer is generated by the Web Server Control event code.

**Notes**

- The Orders table frame is generated on the fly by executing a query to a tcursor, then scanning the tcursor to create an HTML table.
- To see the underlying code, click Form, Design Form when viewing the server, then right-click the server and click Object Explorer. Double-click the OnPostRequest and OnGetRequest events to look at the code. You can see how the query is defined and run to produce the HTML table showing the customer's orders.
- To see the HTML code used in this application, open QUERY.HTM and SQUERY.HTM in WEBSRV\SAMPLE\PAGES. The page with the dynamic list box is generated using HTML snippets borrowed from Sample 3 (SERVER.FSL).

---

**{button ,AL(`APPLETS;',0,"Defaultoverview",)}** **Related Topics**

## Sample 3: SERVER.FSL

This is a sample corporate Internet application that uses a table-driven request handling routine. Requests are mapped to handlers in ObjectPAL 'helper' libraries in SERVER.DB. SERVER.FSL receives HTTP requests, checks the SERVER.DB table, and automatically dispatches requests to the appropriate helper library. This allows several ObjectPAL Internet applications to be running on one server at the same time.

This Web site uses four pages. Two of these pages, HERCULES.HTM and NEWCUST.HTM, can be found in the WEBSRV\SAMPLE\PAGES directory. The other two are generated from the sample helper library, HERCULES.LSL, that contains the code for building these pages. It contains several examples of posting, querying, and a technique of templatized HTML page generation using the HTML snippets stored in HERCULES.DB.

**To run the application**

1. Set the Corel Paradox working directory to WEBSRV\SAMPLE under the Paradox program directory.
2. Double-click SERVER.FSL to run the form containing the Corel Web Server Control.
3. Start your Web browser and specify your machine name or IP address as the location to open.

   The default page is set in the Corel Web Server Control Properties so, when you open your machine location with the form running, the server knows the correct page to send to the browser. (To view the Web Server Control Properties dialog box, right click the Web Server Control in Design mode and click Properties, Corel Web Server Control.)

The Hercules Glass company Home Page is now loaded. From here you can

- enter and submit mailing list information (which will appear in the CONTACTS.DB table in the WEBSRV\SAMPLE directory)
- search the Hercules mailing list. This will prompt the Corel Web Server Control to retrieve information from the CUSTOMER.DB table by running a query. If you have already submitted your name and address, it will appear in the list.
- browse the master mailing list. The list is generated from the CUSTOMER.DB table. If you have already submitted your name and address, it will appear in the list.

**Notes**

- You can add your own helper library to handle your application's HTTP requests by simply mapping the application's URIs to library methods in SERVER.DB. The SERVER.FSL form handles the rest.
- It isn't necessary to shut down the server for maintenance and editing of helper libraries. Just revise code in the library, then click the Unload Helper Libraries button to flush the server's cache of helper libraries. When an HTTP request is received that is mapped to your library, it will be reloaded automatically.
- To see the HTML code used in this application, open HERCULES.HTM and NEWCUST.HTM in WEBSRV\SAMPLE\PAGES. You can also edit the HTML code that generates the master mailing list and search pages by opening the HERCULES.DB table and double-clicking the desired field to edit the contents.

---

**{button ,AL(`APPLETS;',0,"Defaultoverview",)} Related Topics**

# Performance tips

### Saving memory

It is convenient to have many threads waiting for client connections, but this can use a lot of memory. You can use the MinReadyConnections property to control this factor.

### Avoiding bottlenecks

The Corel Web Server Control is multi-threaded, i.e., each client request gets its own thread for processing its request. The server was designed with a built-in server manager to overcome the limitations in the OLE ConnectionPoint model. The server manager queues all connections that need to send to the container for processing, and notifies the container that it has pending requests. The container can then pull from the server manager's request queue when it is ready to handle the request. This allows both single- and multi-threaded containers to design their own scheduling and avoid being flooded with requests.

The server manager also prevents bottlenecks in single-threaded containers by providing Events properties that allow users to specify which HTTP requests should fire events. For example, you could write a Web server application which handles only POST events by setting only the NotifyOnPost property to True. This allows clients requesting the default Web page on a GET request to complete uninterrupted, even if another client request is processing a long query on a POST request.

To take advantage of multi-threaded containers, such as Delphi 2.0 or a C++ application, the Corel Web Server Control surfaces the WaitforRestart property and RestartThread() method in the event's ResponseRecord. This allows the user to write multi-threaded event handlers which do not bottleneck the server. To write a multi-threaded event handler, the user spawns a container-side worker thread each time an event is fired to the container, passing the event's RequestRecord (incoming information) and ResponseRecord (outgoing information). The event handler then toggles the WaitForRestart property and then returns from the event handler. This keeps the client request thread waiting while the worker thread on the container side continues to process the request. Once the container side worker thread is done, it calls RestartThread() method, which notifies the waiting ActiveX control/browser-side request thread to wake up and send the result back to the client browser.

---

**{button ,AL(`USING;',0,"Defaultoverview",)} Related Topics**

# About cookies

Cookies are pieces of persistent information (state objects) included with HTTP response objects. The Corel Web Server Control can use cookies to both store and retrieve information from clients. The cookie includes a description of the range of URLs for which that state is valid. Any future HTTP requests made by the client which fall in that range return the current value of the cookie from the client back to the server.

This simple mechanism enables a host of new types of applications to be written for Web-based environments. For example, shopping applications can now store information about currently selected items, for-fee services can send back registration information and free the client from retyping a user ID on the next connection, and sites can store per-user preferences for clients and have clients supply those preferences every time they connect to that site.

## Cookie methods and properties

A cookie is introduced to the client by including SetCookie as part of an HTTP response. The Corel Web Server Control includes these additional Request and Response methods and properties for handling cookies:

- GetCookieName
- GetCookieByIndex
- GetCookie (in Request)
- GetCookie (in Response)
- NCookies

## Cookie specifications

The Netscape cookie specification is currently available on Netscape's Web site. To view it, open the following URL:

<http://home.netscape.com/newsref/std/cookie_spec.html>

---

**{button ,AL(`ABOUT;',0,"Defaultoverview",)}** **Related Topics**

# About the Corel Web Server Control access log

The access log contains a record of all completed requests to the Corel Web Server Control, regardless of returned HTTP status. Requests that were interrupted by the browser or by network errors are not logged in the access log.

### Activating the log

To start logging server access records, set the Web Server property EnableAccessLogging to True, and enter a log name as the value for the AccessLog property.

You can change these values in the Object Explorer or in the Web Server Properties dialog box (Logging page). (You can log transactions as well as access records by setting the appropriate debug log properties on the Logging page.)

### Format

The access log complies with the NCSA/CERN Common Log Format (CLF). A sample entry is shown below (each entry is on a single line in the actual logfile):

146.82.56.219 - - [23/Apr/1996:05:28:13 -0800] "GET /newinfo/32srvr/newpage.html HTTP/1.0" 200 29764

A dash or minus means null or not applicable. The fields are (in order)

- IP address or host name of the client. For host name logging LookupClientName must be enabled (not recommended).
- Unused (placeholder for RFC931/TAP identification; deprecated, not supported)
- Authenticated user name. This is present only if the browser sends basic authentication header with the request.
- Date/time per Common Log Format (local time, with GMT offset)
- quoted string containing the HTTP request line, including method, URL, and HTTP version fields
- the HTTP/1.0 status/result code for the request
- the number of bytes, exclusive of HTTP header bytes, returned to the client as a result of the request

---

**{button ,AL(`ABOUT;',0,"Defaultoverview",)} Related Topics**

# Corel Web Server Control events and methods

The Corel Web Server Control uses a number of events and methods to perform transactions over the Internet. For details, click the underlined text.

Most of the events involve two OLE automation objects: Request and Response. These are defined according to HTTP specifications.

### Events

- AfterRequest
- DoShutDown
- DoStartUp
- OnGetQuery
- OnGetRequest
- OnHeadRequest
- OnPostRequest

### Methods

- AboutBox
- AddMIMEType
- GetMIMEType
- GetMIMETypeByExt
- RemoveMIMEType
- SetMIMEType
- ToggleActive

# About Corel Web Server Control properties

To set Corel Web Server Control properties, right-click the Corel Web Server Control and click Properties, Corel Web Server Control.

In the Corel Web Server Control Properties dialog box, click a tab to set each kind of property:

- Server properties
- Pages properties
- Logging properties
- MIME properties
- Events properties
- Status properties

There are other properties not available on these pages that you can only set with ObjectPAL code:

- Other properties

---

**{button ,AL(`INTRO_PROP;',0,"Defaultoverview",)}** **Related Topics**

# Corel Web Server Control property list

These properties apply to the Corel Web Server ActiveX control. You can set most of them on the Web Server Control property pages. For more information, see About Corel Web Server Control properties.

- AccessLog
- Active
- AllowKeepAliveConnection
- BasePath
- ConnectionTimeOut
- DebugLog
- DefaultPage
- EnableAccessLogging
- EnableDebugLogging
- FooterPage
- HeaderPage
- HTTPPort
- LookupClientName
- MaxConnections
- MaxKeepAliveRequest
- MinReadyConnections
- NCompletedConnections
- NMIMETypes
- NotifyAfterRequest
- NotifyOnGet
- NotifyOnGetQuery
- NotifyOnHead
- NotifyOnPost
- ShowActiveConnections
- ShowCompletedConnections
- UpdateServerStatusBar

---

**{button ,AL(`INTRO_PROP;',0,"Defaultoverview",)} Related Topics**

## Other Web Server properties (unsurfaced)

The following property isn't available through a property page, but you can still set it with ObjectPAL:

- Active

---

**{button ,AL(`INTRO_PROP;',0,"Defaultoverview",)} Related Topics**

# Request object

The Request OLE Automation object represents an in-bound communication to the Corel Web Server Control from a client browser. Its form and contents are compliant with Hypertext Transfer Protocol (HTTP) 1.0 and meet most of the requirements for version 1.1, as defined by the Internet Engineering Task Force (IETF). For the latest protocol draft, see:

<URL:http://www.ics.uci.edu/pub/ietf/http/>

The following read-only properties and methods identify the sender and include instructions for the return communication (Response object) plus the request content and any authentication required for acceptance of the request:

- Accept
- AcceptCharset
- AcceptEncoding
- AcceptLanguage
- AuthorizationPassword
- AuthorizationUserid
- ClientName
- Content
- ContentLength
- Date
- Extra
- From
- GetFieldName
- GetFieldByIndex
- GetField
- GetCookieName
- GetCookieByIndex
- GetCookie
- Host
- HTTPVersion
- IPAddress
- IfModifiedSince
- NFields
- NCookies
- Pragma
- QueryString
- Referer
- URI
- UserAgent

# Response object

The Response OLE Automation object represents an out-bound communication from the Corel Web Server Control to a client browser. Its form and contents are compliant with Hypertext Transfer Protocol (HTTP) 1.0 and meet most of the requirements for version 1.1, as defined by the Internet Engineering Task Force (IETF). For the latest protocol draft, see:

<URL:http://www.ics.uci.edu/pub/ietf/http/>

The following Response method is used in multi-threaded containers to restart a thread that was put on hold:

- RestartThread

These Response methods are used to send and read "cookies":

- GetCookie
- SetCookie

The following Response properties provide information about the response and communicate its content:

- Allow
- ConnectionKeepAlive
- ContentEncoding
- ContentType
- Date
- Expires
- Extra
- FooterFile
- HeaderFile
- LastModified
- Location
- NBytesSent
- ResultFile
- ResultString
- Server
- StatusCode
- WWWAuthenticateRealm
- WaitForRestart

# Corel Paradox Internet tools glossary

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

**A**

ActiveX control

**B**

BDE

browser

**C**

client

cookies

**D**

DLL

dynamic

**E - F**

**G**

GXEngine

GXPublish

**H**

HTML

**ActiveX control**

An independent object created using technology that allows the control to be used by different applications. ActiveX controls may be as simple as a button or as complex as complete applications and can be developed in a variety of programming languages.

**BDE (Borland Database Engine)**

Borland Database Engine (formerly IDAPI). This application uses this database engine to access and deliver data. BDE maintains information about your PC's environment in the BDE configuration file (usually called IDAPI32.CFG).

**browser**

An application used to view HTML documents from the Internet.

**client**

The user requesting information from a server.

**cookies**

Messages sent between Web browsers and Web servers which allow the browser to maintain information about a client and allow the server to customize responses. For example, a cookie might tell the server which pages you viewed the last time you visited that website. The server could then present you with the same pages on subsequent visits.

**DLLs (Dynamic Link Libraries)**

Files needed by applications to perform various functions, called by a program as needed. Some .DLLs are common to all Windows applications; others are application-specific.

**dynamic**

A document that is produced on request so that it always contains the most up-to-date information.

**GXEngine**

A tool used by the Corel Web Server control and the Web Server Repository to output BDE-based data into HTML format.

**GXPublish**

A tool used by the Corel Web Server control and the Web Server Repository to dynamically publish BDE-based data into HTML format.

**HTML (HyperText Markup Language)**

The programming language used to create documents for the Internet.

**HTML Publishing Library**

A set of custom ObjectPAL methods and types which can be used by programmers to access Corel Paradox static and dynamic HTML publishing features.

**HyperText template (HTT) files**

Files used for producing dynamic HTML documents. The template is combined with the requested data to produce an HTML file to send back to the browser.

**HTTP (HyperText Transfer Protocol)**

The protocol used by browsers to get documents from Web servers over the Internet.

**hyperlink**

A text string or graphic element in an HTML document which is linked to another document (or a different place in the same document). Hyperlinks are usually set apart from regular text or graphics by colored underlining or a border.

**library**

A Corel Paradox object that stores custom ObjectPAL code. Libraries are useful for storing and maintaining frequently used routines and for sharing custom methods and variables among forms, scripts, and other libraries.

**MIME (Multipurpose Internet Mail Extensions)**

A specification for transmitting different file formats over the Internet.

**multi-threading**

A program’s ability to execute several different tasks simultaneously.

**ObjectPAL**

The Corel Paradox programming language.

**OLE object**

An object created in one application and used in another by means of OLE (object linking and embedding). OLE objects are connected to the application in which they were created and can be modified from within the application in which they have been placed.

**Request object**

The Request OLE Automation object represents an in-bound communication to the Corel Web Server Control from a client browser.

**Response object**

The Response OLE Automation object represents an out-bound communication from the Corel Web Server Control to a client browser.

**static**

A document that represents data at a given point in time and is not updated when requested by a browser.

**TCP/IP (Transport Control Protocol/Internet Protocol)**

Standard protocol for transferring data over networks.

**URL (Uniform Resource Locator)**

Address of a document or other resource on the Internet. Used by browsers to locate information such as home pages or downloadable files.

**Web server**

A computer used to receive requests from browsers and deliver the appropriate response (e.g., a Web page).

**Web Server Repository**

Corel Paradox storage facility for HTML template files created by publishing files to HTML.

# Corel Paradox Internet tools basic tasks

The Corel Paradox Internet tools allow you to harness the power of the Internet with a few simple steps. (Use the Corel Web Server (.EXE) for simple, straightforward Web publishing. Use the Corel Web Server Control if you want to program the server to send custom responses to browser requests or process information posted by electronic forms.)

### Step 1: Creating and working with HTML documents

You can easily publish reports and tables to the Internet using the Corel Paradox HTML Publishing Experts to translate them into HTML. You can also create static HTML documents from Paradox forms. For more information, see the following topics:

- Publishing reports to HTML
- Publishing tables to HTML
- Publishing forms to HTML

### Step 2: Preparing to use the Corel Web Servers

You must have a TCP/IP network connection for the Corel Web Servers to function properly. For more information, see the following topic:

- Installing a TCP/IP protocol

You must also have an properly configured Web server running on your system in order to receive and process browser requests.

- Setting up the Web server
- Setting Web server properties

### Step 3: The Web server in action

- Starting the Web server
- Testing the Web server
- Viewing Web server connections

---

**{button ,AL(`BASIC;',0,"Defaultoverview",)}** **Related Topics**

## Publishing reports to HTML

The Corel Paradox HTML Report Expert easily translates any Corel Paradox report to an HTML file, allowing you to choose from a number of publishing options as you go.

**To publish a report to HTML**

1. View the report.
2. Click File, Publish to HTML.
3. Follow the steps in the HTML Report Expert.

**Note**

- The Corel Web Servers are case-sensitive, so make sure you note the exact filename used when saving documents for future reference.

---

**{button ,AL(`DOCUMENT;LIST;',0,"Defaultoverview",)}** **Related Topics**

## Publishing tables to HTML

The Corel Paradox HTML Table Expert easily translates any Corel Paradox table to an HTML file, allowing you to choose from a number of publishing options as you go.

**To publish a table to HTML**

1. View the table.
2. Click File, Publish to HTML.
3. Follow the steps in the HTML Table Expert.

**Note**

- The Corel Web Servers are case-sensitive, so make sure you note the exact filename used when saving documents for future reference.

---

**{button ,AL(`DOCUMENT;LIST;',0,"Defaultoverview",)}** **Related Topics**

## Publishing forms to HTML

You can publish a Corel Paradox form to a static HTML document. This feature works best with simple forms that use text, edit boxes, list boxes, radio buttons or check boxes. Form objects such as graphics, table frames, crosstabs, notebooks and charts do not translate statically to HTML.

**To publish a form to an HTML file**

1. View the form.
2. Click File, Publish to HTML.
3. Type a filename.
4. Choose .HTM as the file type from the drop-down list.
5. Click the Save button.

Corel Paradox saves an HTML version of your form that can be viewed by a Web browser.

**Note**

- Corel Paradox automatically adds FORM METHOD and ACTION tags to any form published to HTML. By default the FORM METHOD tag is set to POST and the ACTION is set to the Paradox form object's noise name (e.g., #Form1). To set these properties yourself, change the HTMLMethod property or the HTMLAction property of the form using the Object Explorer.
- Paradox also adds a Submit button to static forms published to HTML. For the Submit button to work with the Corel Web Server Control, you must add code to trap the POST action in the OnPostRequest event. For more information, see Processing information from HTML input forms.
- The Corel Web Servers are case-sensitive, so make sure you note the exact filename used when saving documents for future reference.

---

**{button ,AL(`DOCUMENT;LIST;',0,"Defaultoverview",)} Related Topics**

## Installing a TCP/IP protocol

Your machine must have a TCP/IP protocol installed for the Corel Web servers to function properly.

**To install the TCP/IP protocol on your system**

- Consult your Windows documentation

---

**{button ,AL(`SETUP;LIST;',0,"Defaultoverview",)}** **Related Topics**

## Setting up the Web server

The Corel Web Server (.EXE) and related documentation are included when you install Corel Paradox. The Setup program also copies the Corel Web Server Control (WEBSRV.OCX) and supporting DLLs into appropriate directories and registers them.

**To set up the Corel Web Server (.EXE)**

- Choose Corel Web Server from the Start menu (by default, it is placed in the Accessories submenu of the Corel Paradox 8 program group). The Corel Web Server icon will appear on the Taskbar. It is active and ready to receive browser requests.

For help using the Corel Web Server, right-click the Corel Web Server icon and click View connections to maximize the Web server window. Choose Help Topics from the Help menu to open the Corel Web Server (.EXE) Help file.

**To set up the Corel Web Server Control**

- Place a Corel Web Server Control object on a Corel Paradox form. For more information, see the following topics:
  - Adding the Corel Web Server Control Toolbar button
  - Adding the Corel Web Server Control to a form

---

**{button ,AL(`SETUP;LIST;',0,"Defaultoverview",)} Related Topics**

## Adding the Corel Web Server Control button to the toolbar

**To add the Corel Web Server Control button to the toolbar**

1. Create a new blank form or open an existing form in Design mode.
2. Click View, Toolbars and enable the Object check box.
3. Right-click within the Object toolbar in a blank area and click Add ActiveX Control.
4. Choose Corel Web Server Control in the Insert Control dialog box.

When you click the OK button, this button appears on the ActiveX Controls tab of the Object toolbar and the alternate ActiveX Controls view of the Standard toolbar:

---

**{button ,AL(`SETUP;LIST;',0,"Defaultoverview",)}** **Related Topics**

## Adding the Corel Web Server Control to a form

**To add the Corel Web Server Control to a form**

1. Open a form in the Form Design window.
2. Click the Corel Web Server Control button on the Object toolbar.
3. Click and drag to draw the Corel Web Server Control window.

---

**{button ,AL(`SETUP;LIST;',0,"Defaultoverview",)}** **Related Topics**

## Setting Web server properties

You can set properties for the Corel Web Servers using the appropriate Web server Properties dialog box. The Base Path and Default Page properties on the Web server Properties dialog box Pages page must be set for the Web server to function properly.

**To set Corel Web Server (.EXE) properties**

- Right-click the Corel Web Server icon on the Taskbar and click Properties to open the Web server Properties dialog box.

For more information about Corel Web Server properties, see the Corel Web Server (.EXE) Help file.

**To set Corel Web Server Control properties**

- Right-click the Corel Web Server Control object and choose Properties... Corel Web Server Control to open the Web server Properties dialog box.

For more information about Corel Web Server Control properties, see About Corel Web Server Control properties.

---

**{button ,AL(`SETUP;LIST;',0,"Defaultoverview",)} Related Topics**

## Starting the Web server

The Web server must be active in order to process requests from browsers. The Corel Web Server (.EXE) is launched in an active state, however, it can be shut down without exiting the server.

**To start the Corel Web Server (.EXE)**

- If the Corel Web Server (.EXE) has been shut down, you can either click the Startup/Shutdown toggle toolbar button, or right-click the Corel Web Server icon on the Taskbar and click Startup.

**To start the Corel Web Server Control**

- Run a form containing a Corel Web Server Control object.

When you start the Corel Web Server Control, you have a fully functional Web server that can retrieve static HTML documents and graphics.

---

**{button ,AL(`ACTION;LIST;',0,"Defaultoverview",)}** **Related Topics**

## Testing the Web Server

You can test the Corel Web Server (.EXE) or the Corel Web Server Control using your machine as both the client and the server.

**To test the Web server**

1. Launch your Web browser.
2. Type your computer name as the address to locate.

   If you don’t know your computer name, consult your Windows documentation.

The page you specified in the Web server Base Path property will be displayed by the browser.

**Note**

- The Corel Web Servers are case-sensitive. Be sure to use the exact same filename you specified in the HTML Publishing Expert.

---

**{button ,AL(`ACTION;LIST;',0,"Defaultoverview",)}** **Related Topics**

## Viewing Web server connections

Browser requests are displayed in the Corel Web Server (.EXE) window and in the Corel Web Server Control object on a form.

**To view connections to the Corel Web Server (.EXE)**

- Maximize the Corel Web Server window by right-clicking the Corel Web Server icon on the Taskbar and clicking View Connections.

**To view connections to the Corel Web Server Control**

- View the Corel Paradox form containing the Corel Web Server Control object.

---

**{button ,AL(`ACTION;LIST;',0,"Defaultoverview",)}** **Related Topics**