ABOUT THIS GUIDE 7
Common conventions 7
The File Manager 7
Reporting Problems 7
STYLE GUIDE 8
DATABASE RECORD DISPLAY 9
THE BASICS 9
Text field options: 9
LIMIT 9
ENCODING 9
Image field options: 9
URL 9
Date field options: 9
DATE FORMAT 9
Number field options: 10
PRECISION 10
Linking: 10
LINK 10
Advanced options: 10
RECORD ID 10
STACK 10
MANUAL ID 10
Deleting A Record 11
DELETE WITH ICON: 11
DELETE WITH LINK 11
DELETE WITH AN IMAGE 11
RECORD OWNER DATA 11
WORKFLOWS/FORM FIELD DATA 12
Displaying Values 12
TEXT/NUMBER FIELDS 12
WORKFLOWS/DATABASE RECORD ENTRY 13
System Fields 13
SYSTEM ERROR 13
HIDE SHOW Record 13
RECORD NAME 13

RECORD DESCRIPTION *13*
Form Controls 13
SUBMIT BUTTON *13*
LOADING GRAPHIC *13*
Fields 14
INPUT *14*
LABEL *14*
INSTRUCTIONS *14*
REQUIRED IMAGE *14*
Text field 14
COUNTER *14*
Image field 14
OLD *14*
ALT TEXT *14*
LINK *14*
Video field 15
OLD *15*
DESCRIPTION REQUIRED *15*
DESCRIPTION FIELD *15*
WIDTH/HEIGHT REQUIRED *15*
WIDTH/HEIGHT FIELD *15*
WIDTH/HEIGHT DESCRIPTION *15*
Date field 15
YEAR *15*
MONTH *15*
DAY *15*
Downloadable/Audio 16
OLD *16*
VARIABLES 17
SETTING VARIABLES 17
Example: Setting variables 17
MANUALLY *17*
RECORD FIELD *17*
RECORD ID *17*
RECORD OWNER *17*
FORMULA VALUE *17*
OPTIONS 18

OVERWRITE *18*
QUICK COPY 18
APPEND 18
SETSTACK 18
E-COMMERCE - PRODUCTS AND DEPARTMENTS 19
Detail and Department Template pages: 19
Text Fields 19
PRODUCT NAME *19*
PRODUCT SUBTITLE *19*
PRODUCT DESCRIPTION *20*
PRODUCT SHORT DESCRIPTION *20*
PRODUCT COMMENTS *20*
PRODUCT PRE-SALE PRICE *20*
PRODUCT PRICE (simple) *20*
PRODUCT PRICE (advanced) *20*
PRODUCT ITEMS *20*
PRODUCT STATUS *20*
PRODUCT SKU *20*
PRODUCT URL *21*
PRODUCT KEY *21*
Images 21
PRODUCT DETAIL *21*
PRODUCT THUMBNAIL *21*
PRODUCT INDEX *21*
Form Elements 21
PRODUCT ITEM DROP-DOWN *21*
PRODUCT QUANTITY INPUT *21*
ADD TO CART BUTTON *21*
CHECKOUT BUTTON *22*
Miscellaneous 22
CUSTOM FIELDS *22*
EDIT TOOL *22*
Conditional Display 22
PRICING product_quantity_discount: *22*
PRICING Product_on_sale: *22*
PRICING product_items: *22*
Departments 23

Properties 23
NAME 23
DESCRIPTION 23
HEADER 23
IMAGE 23
KEY 23
Links 23
URL 23
LINK 23
Miscellaneous E-Commerce 23
CART ITEM COUNT 23
E-COMMERCE - SHOPPING CART 24
Shopping Cart Summary 24
TOTAL ORDER 24
SALES TAX 24
SHIPPING 24
HANDLING 24
SUBTOTAL 24
CONTINUE SHOPPING BUTTON 24
UPDATE TOTALS BUTTON 24
CHECKOUT BUTTON 24
Shopping Cart Items 24
THUMBNAIL 24
ITEM NAME 25
DESCRIPTION 25
STATUS 25
QUANTITY VALUE 25
QUANTITY INPUT 25
PRICE 25
SUBTOTAL 25
DELETE BUTTON 25
Shopping Cart Coupon Management 25
DESCRIPTION 25
SAVINGS 25
DELETE BUTTON 25
ADD BUTTON 26
COUPON FIELD 26

VARIABLE: coupon_set 26

E-COMMERCE - PAYMENT FORM 27

Elements 27

EDIT ADDRESS LINK 27

BILLING ADDRESS 27

SHIPPING ADDRESS 27

SHIPPING FORM ELEMENT 27

E-COMMERCE - RECEIPT PAGE 28

Elements 28

ORDER NUMBER 28

BILLING ADDRESS 28

SHIPPING ADDRESS 28

CARD NUMBER 28

CARD EXPIRATION 28

COUPON CODE 28

DYNAMIC DATA DISPLAY 29

FORMULA VALUE 29

REPETITION COUNT: 29

REMAINING REPETITIONS: 29

REPETITION COUNT 29

CUSTOM LOGIN FORM DISPLAY 30

Fields 30

PASSWORD 30

USER ID 30

SUBMIT 30

USER INFORMATION 31

HANDLE/USER ID 31

FIRST AND LAST NAME 31

HANDLE WITH FIRST/LAST NAME 31

EMAIL ADDRESS 31

USER KEY 31

FIRST NAME 31

LAST NAME 31

CURRENT DATE 32

OUTPUT: January 20, 2008 32

OUTPUT: 1-20-08 ..... *32*
OUTPUT: 08 ..... *32*
OUTPUT: 2008 ..... *32*

FILE MANAGER **33**
ECHO FILE PATH *33*

SESSION **34**
ENCRYPTED SESSION ID *34*
USER IP ADDRESS *34*

USER LOCATION **35**
COUNTRY CODE *35*
COUNTRY CODE 3 LETTERS *35*
COUNTRY NAME *35*
REGION *35*
CITY *35*
ZIP *35*
LATITUDE *35*
LONGITUDE *35*
METRO CODE *35*
AREA CODE *35*
COUNTY *35*
USER IP ADDRESS *35*

About This Guide

This guide is intended to introduce you to one of the most powerful features of HyperSites Builder… we call it HyperCode. While we advertise that no programming is required, some may view HyperCode as programming-esque. It is more like inserting fields into mail merge in Microsoft Word, or inserting data into a FileMaker "layout" within the database.

Here's an example:

```
<[echo_data(type="record" field="vehicle_name")]>
```

This will take the value from the field "vehicle_name" in the current record and insert it into the document.

If you're using a Text Content window, the text may look like this:

> *At John Barret Ford, we're committed to providing the best price on <[echo_data (type="record" field="vehicle_name")]>s in the entire state.*

When the text is displayed in the page, it will look like this:

> *At John Barret Ford, we're committed to providing the best price on Mustangs in the entire state.*

Common conventions

The File Manager

Some of our HyperCode makes reference to files in your file manager. When working with these fields, here are some simple guidelines to follow.

THE PATH: When referring to the path to the file in the File Manager, you should start the path with forward slash. For example, `/file.jpg` If the file is in a directory in the file manager, reference it the following way: `/directory/file.jpg`

Reporting Problems

If you notice typographical errors or have any questions at all, feel free to let us know at support@hypersites.com

Style Guide

A `mono-spaced` font represents the HyperCode itself.

Bolded text is data or content you provide: **`product`**

Italics define optional parameters that are not required for the display to work: *`stack="`**`number`**`"`*

The pipe character | indicates that there are multiple options for a given parameter, and only one should be used. `scope="page|site"`

Database Record Display

The Database Manager is a powerful system that gives your sites truly dynamic features. Creating databases and entering data is simple enough. But if you want more control over the data from the databases than using simple record display content, using HyperCode is just what you need.

THE BASICS

To get data from a Database Record into your page use the following:

```
<[echo_data(type="record" field="field name")]>
```

Text field options:

LIMIT
Truncates the text returned from a field to the number of characters specified. For example, if the text returned is 259 characters, and you only want the first 25, you would use: `limit="25"`

```
<[echo_data(type="record" field="field name" limit="number")]>
```

ENCODING
Encodes text from the given field into the specified format.

```
<[echo_data(type="record" field="field name" encoding="pn|url|key")]>
```

Option	Description	Source	Result
pn	Converts the text in the given field into a value safe for HyperSites URLs.	The big brown cow	The_big_brown_cow
url	Converts the text in the given field into a value safe for standard URLs.	The big brown cow	The+big+brown+cow
key	Converts the text in the given field into a value safe for HyperSites URL Key.	The big brown cow	The-big-brown-cow

Image field options:

URL
Returns just the URL of the image, and does not return the full <img> tag. This is useful if for some reason you want to build your own image tag. For example, you could specify your own `onClick` target, or you could add style controls to further customize the display.

```
<[echo_data(type="record" field="field name" property="url")]>
```

Date field options:

DATE FORMAT

Provides a simple means to a completely customized date format. For the specific format strings, see http://us.php.net/manual/en/function.date.php . For example, the format string “l, F jS Y” will output a date with the following format: Saturday, October 6th 2007

```
<[echo_data(type="record" field="field name" date_fomat="format")]>
```

Number field options:

PRECISION
Set the number of decimal points to return in the output. If your field has a value of 12.345, and you set `precision="2"`, HyperSites will output 12.34.

```
<[echo_data(type="record" field="field name" precision="number")]>
```

Linking:

LINK
If you've used the Record Detail Container to tell HyperSites that a page can display a given record type, you can easily build a link to that page for current record data. If you're not using the Record Detail Container you can add the optional `page` parameter to specify the page to link to instead.

If you're using URL Keys, HyperSites will automatically determine the proper URL structure and build the link accordingly.

```
<[echo_data(type="record" element="link" page="page id")]>
```

Advanced options:

RECORD ID
When building URLs, you'll need a quick way to get the ID of the current record to pass.

```
<[echo_data(type="record" element="id")]>
```

For example: pg1234-as<[echo_data(type="record" element="id")]>.html would be rendered as pg1234-as1234.html if the current record ID is 1234.

STACK
To work with field data from a record outside of the current level, use the stack. The HyperSites stack is a way to access several layers of field data individually. For example, if you were listing blog entries by author, you would create a Dynamic Data Display to list the authors, then nest another Dynamic Data Display inside of the authors to get their associated blog entries. If you then wanted to include field data from the author's record, you would use `stack="1"`

```
<[echo_data(type="record" field="field name" stack="number")]>
```

MANUAL ID
If you need to echo a value from a specific record you ca specify the ID of a record to use.

```
<[echo_data(type="record" field="field name" id="record id")]>
```

You can also use a variable that was set in another part of the page.

```
<[echo_data(type="record" field="field name" id="var:variable_name")]>
```

If you have attached a database to your products, you can use the variable 'pr' to get those values.

```
<[echo_data(type="record" field="field name" id="var:pr")]>
```

DELETING A RECORD

DELETE WITH ICON:
Provides a HyperSites delete tool for a user to delete a record.

```
<[delete(type="record")]>
```

DELETE WITH LINK
Provides a link for a user to delete a record.

```
<[delete(type="record" text="link text")]>
```

DELETE WITH AN IMAGE
Provides a link for a user to delete a record.

```
<[delete(type="record" image="file")]>
```

The `image="file"` refers to an image in the File Manager. See Common Conventions elsewhere in this document for more details.

RECORD OWNER DATA

You can access the details of the record's owner for display in a page. If, for example, you have a workflow that is only available to users of your site, we provide the details of the user that created the record from the HyperSites Users and Groups system.

The basic HyperCode is:

```
<[echo_data(type="record" element="owner" field="field name")]>
```

The options for "field name" are as follows:

```
first_name, last_name , full_name, email, subcustomer_name, subcustomer_id, company,
handle, password, key, and date_created
```

Workflows/Form Field Data

When working with Workflows, it can be handy to use some of the values in the forms before the Workflow is actually committed. This is great when adding Workflows to the shopping cart or when using Instant Purchase option. You can set the Price, Name, and Description fields using data from the current Workflow.

Displaying Values

TEXT/NUMBER FIELDS

Currently, Text and Number field types are supported. The `step` property is optional and is only useful in multi-step Workflows. If not provided, step defaults to the last step (and usually the only step) in the Workflow.

In addition, only values from Database steps are allowed. Actions are not supported.

```
<[echo_data(type="workflow" field="field name" step="step")]>
```

Workflows/Database Record Entry

Building custom forms for your workflows will make the user experience much better. It takes a little time, but we think it is well worth it. To use the Hypercode below, you must be in a workflow container, with the Customize Display checkbox checked. Once that box is checked, the container will look just like a general container, with the blue plus.

All of the HyperCode in this section returns plain text with no formatting. The input fields have a `css` property so you can customize the display. The syntax for the `css` property is basic inline css:
`width: 300px;background-color:#cccccc;`

System Fields

SYSTEM ERROR
Shows any error that has occurred in the page load, including missing values, field requirement errors, etc.

```
<[echo_dbfield(item="error")]>
```

HIDE SHOW Record
Allows you to toggle the visibility of the record. Only shows once if not already hidden ;-)

```
<[echo_dbfield(item="hideflag" style="css")]>
```

RECORD NAME
Displays the system record name text field

```
<[echo_dbfield(item="name" style="css")]>
```

RECORD DESCRIPTION
Displays the system record description field

```
<[echo_dbfield(item="description" style="css")]>
```

Form Controls

SUBMIT BUTTON
Display the submit button for an asset form. Uses the workflow's settings, with no overrides for images.

```
<[echo_dbfield(item="button" style="css")]>
```

LOADING GRAPHIC
Display the 'loading' graphic for the form.

```
<[echo_dbfield(item="loading")]>
```

Fields

The following field types are rendered, but have no specific "parts" available: bulleted list, selection, number, association, email, country, state, instructions, zip

INPUT
Displays the main entry field for a record field

```
<[echo_dbfield(field="name" style="css")]>
```

LABEL
Displays the Form Label field

```
<[echo_dbfield(field="name" part="label")]>
```

INSTRUCTIONS
Displays the Instruction field

```
<[echo_dbfield(field="name" part="instructions")]>
```

REQUIRED IMAGE
Displays the required/missing arrow. The `required` property allows you to override whether the field is required in this form.

```
<[echo_dbfield(field="name" part="required" required="yes|no")]>
```

Text field

COUNTER
Displays the counter for the remaining characters available

```
<[echo_dbfield(field="name" part="counter" style="css")]>
```

Image field

OLD
Displays the link to display the previously uploaded image (if applicable), and the delete tool. Styles apply to the link.

```
<[echo_dbfield(field="name" part="old" style="css")]>
```

ALT TEXT
Displays the alt tag text field.

```
<[echo_dbfield(field="name" part="alt" style="css")]>
```

LINK
Displays the link/destination field.

```
<[echo_dbfield(field="name" part="link" style="css")]>
```

Video field

OLD
Displays the link to display the previously uploaded video (if applicable), and the delete tool. Styles apply to the link.

```
<[echo_dbfield(field="name" part="old")]>
```

DESCRIPTION REQUIRED
Displays the required arrow for the description (not allowed to change required state for this one).

```
<[echo_dbfield(field="name" part="description_required")]>
```

DESCRIPTION FIELD
Displays the description entry field.

```
<[echo_dbfield(field="name" part="description" style="css")]>
```

WIDTH/HEIGHT REQUIRED
Displays the required arrow for the width/height (not allowed to change required state for this one).

```
<[echo_dbfield(field="name" part="width_required|width_required")]>
```

WIDTH/HEIGHT FIELD
Displays the width/height entry field.

```
<[echo_dbfield(field="name" part="width|height" style="css")]>
```

WIDTH/HEIGHT DESCRIPTION
Displays the constraints placed upon the width/height entry field.

```
<[echo_dbfield(field="name" part="width_text|height_text")]>
```

Date field

NOTE: Not using a part will attempt to render the entire date - year, month, and day inline. This will fail if any of the parts have been rendered. This actually enables piecemeal date edits. Only edit the month, or the year, for example. The rest of the date is still there, but it's a step closer to a 'partial' date.

YEAR
Displays the year field.

```
<[echo_dbfield(field="name" part="year" style="css")]>
```

MONTH
Displays the month field.

```
<[echo_dbfield(field="name" part="month" style="css")]>
```

DAY
Displays the day field.

```
<[echo_dbfield(field="name" part="day" style="css")]>
```

Downloadable/Audio

OLD
Displays the file that is already uploaded (if there is one).

```
<[echo_dbfield(field="name" part="old")]>
```

Variables

Like any other web development system, HyperSites provides the ability to set and retrieve variables. Variables can be set at the system, site, and page level and are a fantastic way to improve the dynamic functionality of your site.

SETTING VARIABLES

Setting variables in HyperCode is almost identical to using `<[echo_data()]>`. The difference between `<[echo_data()]>` and setting variables command is the presence of the `variable="name"` parameter. For example, the HyperCode to echo a record field named "Author" is `<[echo_data(type="record" field="Author")]>` To set a variable with the value in the record field "Author" you would use this: `<[set(variable="some_name" type="record" field="Author" scope="page")]>`

Then to use that value, you use echo_data: `<[echo_data(type="variable" name="some_name")]>`

Another difference is the scope parameter. Scope allows you to specify how long the variable will last in memory. The options are "page" and "site".

Example: Setting variables

MANUALLY
Using manually entered variable data:

```
<[set(variable="variable_name" value="variable value" scope="page|site")]>
```

RECORD FIELD

```
<[set(variable="variable_name" type="record" field="field name" scope="page|site")]>
```

RECORD ID

```
<[set(variable="variable_name" type="record" element="id" scope="page|site")]>
```

This will allow you to later use data from the record without first having to "get" the data using a dynamic data display container. If set within a dynamic data display container, the value will be replaced for each instance. Syntax for using an ID in an echo_data is nearly identical:

```
<[echo_data(type="record" field="field_name" id="var:variable_name")]>
```

RECORD OWNER

```
<[set(variable="variable_name" type="record" element="owner" field="full_name"
scope="page|site")]>
```

FORMULA VALUE
in a Repeating Content container at stack 1

```
<[set(variable="variable_name" type="variable" class="record" name="formula name"
stack="1")]>
```

OPTIONS

OVERWRITE

Specifies what to do if the variable value changes:

```
<[set(variable="variable_name" value="variable value" scope="page|site"
overwrite="always|never|update")]>
```

Option	Description	Old Value	New Value	Result
always	Always overwrite the value	Cow	Horse	Horse
		Cow	<no value>	<no value>
never	Once the value is set, never change it	Cow	Dog	Cow
		Cow	<no value>	Cow
update	If a new value is supplied, and isn't empty, update the value	Cow	Horse	Horse
		Cow	<no value>	Cow

QUICK COPY

```
<[echo_data(type="variable" name="variable_name")]>
```

APPEND

To append data to a variable, the usage is identical to set:

```
<[append(variable="some_name" type="record" field="field" scope="page|site")]>
```

SETSTACK

To set the variable value in a particular stack position, use the setstack parameter. The following example sets the variable "as" at stack "3" to "variable_value"

```
<[append(variable="as" value="variable_value" setstack="3" scope="page|site")]>
```

E-Commerce - Products and Departments

HyperCode can be used to achieve custom formatting of e-commerce related pages. It can also be used to show store information on non-e-commerce pages, great for featuring products on your home page, or showing the number of items in the shopping cart throughout your site.

The following code can be used as-is in the Product Detail Container to display information about the product ID passed on the URL using the `pr` variable. For example, to insert the current product's name into the page, you would use the following Hypercode: `<[echo ('store_product_detail','name')]>`

Similarly, to insert the Add to Cart button into the product detail page, you would use the following HyperCode: `<[echo('store_product_detail','form_element','add_button')]>`

This works because the form that gets submitted when the button is clicked is built in to the Product Detail Container. If you were to use this HyperCode in a Department display, it wouldn't work, because the form needed to make the button work is missing.

If you want to put the Add to Cart button in the Department display, instead you would use `add_button_standalone` which wraps the button in the necessary form. In addition, this HyperCode sets the quantity to 1 so that when the user arrives in the cart, there is a quantity to do the math with.

There are cases where you may want to show product details on a page outside of the product and department containers. To do this, we've provided an alternative to `store_product_detail` called `store_product`. The syntax is nearly identical, with one notable exception being the addition of the `product_id` parameter. The `product_id` parameter is where you specify the product's ID as shown in Toolbar > E-Commerce > Store Map.

To show the name of product 122 on the About Us page of your site, you would use the following HyperCode: `<[echo('store_product','122','name')]>` HyperCode that doesn't support this format will be noted. Otherwise it is safe to assume that it is supported.

In summary, to put product details on any page, change `store_product_detail` to `store_product` and insert the product's ID. So, to insert the product's name, `<[echo('store_product_detail','name')]>` becomes `<[echo('store_product','122','name')]>`

Detail and Department Template pages:

Text Fields

PRODUCT NAME
```
<[echo('store_product_detail','name')]>
```

PRODUCT SUBTITLE
```
<[echo('store_product_detail','subtitle')]>
```

PRODUCT DESCRIPTION
```
<[echo('store_product_detail','description')]>
```

PRODUCT SHORT DESCRIPTION
```
<[echo('store_product_detail','short-description')]>
```

PRODUCT COMMENTS
```
<[echo('store_product_detail','comments')]>
```

PRODUCT PRE-SALE PRICE
```
<[echo('store_product_detail','pre_sale_price')]>
```

This code only returns a value if the product's status is set to "On Sale" and the previous price field has a value. You can use this code in conjunction with `<[echo_if('product_on_sale','true text','false text')]>` to show a $ or other text if the product is on sale.

PRODUCT PRICE (simple)
```
<[echo('store_product_detail','price','style')]>
```

`style` is a style name, as defined in Toolbar > Styles > Text Styles

PRODUCT PRICE (advanced)
```
<[echo('store_product_detail','price','style1;style2;style3;style4','table_map')]>
```

The `style` parts are style names, as defined in Toolbar > Styles > Text Styles

`style1` = The numerical price itself

`style2` = No price available error (if the product has no price, for example, 'call to order' products)

`style3` = Quantity pricing value ranges

`style4` = Quantity pricing From, To, and Prices headers.

To show the quantity discount chart for products with quantity pricing, enter 1 for `table map`, 0 for no table (shows the highest quantity discount price.)

If no others styles are specified, `style1` is the default for all style uses. If only `style1` and `style2` are specified, `style1` is used for `style3`. If only `style1` and `style2` are specified, `style2` is used for `style4`.

PRODUCT ITEMS
```
<[echo('store_product_detail','form_element','item_selector')]>
```

This code appears under both 'Text' and 'Form Elements,' since it will show text if there is only one item choice and will show a drop-down menu if there is more than one item choice. Items are size and/or color.

PRODUCT STATUS
```
<[echo('store_product_detail' ,'status')]>
```

PRODUCT SKU
```
<[echo('store_product_detail', 'sku')]>
```

PRODUCT URL
Returns the URL to the product detail page in a department display, ready for use in an href or in the link assistant

```
<[echo('store_product_detail','url','product_detail_page')]>
```

where `product_detail_page` is the ID of the product detail page.

For a link to the product detail page from anywhere on the site, you'll need to supply some additional details.

```
<[echo('store_product','product','url','product_detail_page','department')]>
```

Where `product_detail_page` is the ID of the product template page as shown in Toolbar > Site Tools > Site Map, and `department` is the ID of the department as shown in Toolbar > E-Commerce > Store map.

PRODUCT KEY

```
<[echo('store_product_detail','key')]>
```

To build pretty URLs you'll need to convert product names to text that is unique, and formatted properly.

Images

PRODUCT DETAIL
Intended for product detail pages.

```
<[echo('store_product_detail','image','detail')]>
```

PRODUCT THUMBNAIL
Intended for department product lists.

```
<[echo('store_product_detail','image','thumbnail')]>
```

PRODUCT INDEX
Intended for the store shopping cart.

```
<[echo('store_product_detail','image','index')]>
```

Form Elements

PRODUCT ITEM DROP-DOWN

```
<[echo('store_product_detail','form_element','item_selector')]>
```

This code appears under both 'Text' and 'Form Elements,' since it will show text if there is only one item choice and will show a drop-down menu if there is more than one item choice. Items are any combination of size, color, and width.

PRODUCT QUANTITY INPUT

```
<[echo('store_product_detail','form_element','quantity')]>
```

ADD TO CART BUTTON

Displays a button that when clicked will add the current product to the cart. Intended for use inside of the product display container. This code does not support the manual specification of the product id.

```
<[echo('store_product_detail','form_element','add_button','path to image')]>
```

In a department list, use the following instead. This code includes the form wrapper to make the button itself work. You can specify the product manually, as well.

```
<[echo('store_product_detail','form_element','add_button_standalone','<image file>')]>
```

CHECKOUT BUTTON

```
<[echo('store_product_detail','form_element','checkout_button','path to image')]>
```

In both cases, `path to image` is an optional tag containing the name of a file in the File Manager. See Common Conventions elsewhere in this document for more details.

Miscellaneous

CUSTOM FIELDS

```
<[echo('store_product_detail','custom','field name')]>
```

Where `field name` is the HyperCode Reference assigned in Custom Field Setup in Toolbar > E-Commerce > Store Setup.

EDIT TOOL

If the current user has proper privileges, this will provide a green edit tool.

```
<[echo('store_product_detail','edit_tool')]>
```

Conditional Display

PRICING product_quantity_discount:

```
<[echo_if('product_quantity_discount','true_value','false_value')]>
```

Where `true_value` is the text you would like to show if the condition is met, and `false_value` is the text your want to display if the condition is false.

PRICING Product_on_sale:

```
<[echo_if('product_on_sale','true_value','false_value')]>
```

Where `true_value` is the text you would like to show if the condition is met, and `false_value` is the text your want to display if the condition is false.

PRICING product_items:

```
<[echo_if('product_items','true_value','false_value')]>
```

Where `true_value` is the text you would like to show if the condition is met, and `false_value` is the text your want to display if the condition is false.

Departments

The following HyperCode can be used anywhere there is a `de` variable in the URL to display the properties of a department.

Properties

The Department Property HyperCode will also take a Department ID which will show the value requested for the given department instead of the default department. This is especially handy when you are building a display that is on a regular page or in a Rollover/Image link.

NAME
```
<[echo('store_department_detail','name', 'department')]>
```

DESCRIPTION
```
<[echo('store_department_detail','description', 'department')]>
```

HEADER
```
<[echo('store_department_detail','header', 'department')]>
```

IMAGE
```
<[echo('store_department_detail','image', 'department')]>
```

KEY
Builds a Simple URL friendly version of the current department's name.

```
<[echo('store_department_detail','key')]>
```

To build pretty URLs you'll need to convert department names to text that is unique, and formatted properly.

Links

URL
This will build a URL to the Store Department Display Container if the optional values aren't specified. You can set the `page` variable to `auto` and still specify the `department`.

```
<[echo('store_department_detail','url','page','department')]>
```

LINK
This will build a URL to a Dynamic Data Display Container if the optional values aren't specified.

```
<[echo('store_department_detail','link','page','department')]>
```

Miscellaneous E-Commerce

CART ITEM COUNT
Use this code anywhere on the site to display the number of items in the user's shopping cart.

```
<[echo('store_cart_count')]>
```

E-Commerce - Shopping Cart

Using the HyperCode below, you can build a custom display for your shopping cart. You have complete design control. To use a custom display, check the "Customize Display" option in the carts container setup.

Shopping Cart Summary

TOTAL ORDER
Includes tax, shipping, etc.

```
<[echo('shopping_cart','grand_total')]>
```

SALES TAX

```
<[echo('shopping_cart','sales_tax')]>
```

SHIPPING

```
<[echo('shopping_cart','shipping')]>
```

HANDLING

```
<[echo('shopping_cart','handling')]>
```

SUBTOTAL

```
<[echo('shopping_cart','subtotal')]>
```

CONTINUE SHOPPING BUTTON

```
<[echo('shopping_cart','continue_button','path to file')]>
```

Where `path to file` is an optional path to an image in the File Manager. See Common Conventions elsewhere in this document for more details.

UPDATE TOTALS BUTTON

```
<[echo('shopping_cart','update_button','path to file')]>
```

Where `path to file` is an optional path to an image in the File Manager. See Common Conventions elsewhere in this document for more details.

CHECKOUT BUTTON

```
<[echo('shopping_cart','checkout_button','path to file')]>
```

Where `path to file` is an optional path to an image in the File Manager. See Common Conventions elsewhere in this document for more details.

Shopping Cart Items

THUMBNAIL

```
<[echo('shopping_cart','item','picture')]>
```

ITEM NAME
```
<[echo('shopping_cart','item','item_name')]>
```

DESCRIPTION
```
<[echo('shopping_cart','item','description')]>
```

STATUS
```
<[echo('shopping_cart','item','status')]>
```

QUANTITY VALUE
```
<[echo('shopping_cart','item','quantity')]>
```

QUANTITY INPUT
```
<[echo('shopping_cart','item','quantity_field')]>
```

PRICE
```
<[echo('shopping_cart','item','price')]>
```

SUBTOTAL
```
<[echo('shopping_cart','item','subtotal')]>
```

DELETE BUTTON
```
<[echo('shopping_cart','item','delete_button','path to file')]>
```

Where `path to file` is an optional path to an image in the File Manager. See Common Conventions elsewhere in this document for more details.

Shopping Cart Coupon Management

DESCRIPTION

Displays the coupon's description as entered in Coupon Setup.

```
<[echo('shopping_cart','coupon','description')]>
```

SAVINGS

Displays the coupon's total savings as affected by items in the Shopping Cart.

```
<[echo('shopping_cart','coupon','savings')]>
```

DELETE BUTTON

Allows the user to delete the Coupon they've entered.

```
<[echo('shopping_cart','coupon','delete_button','path to file')]>
```

Where `path to file` is an optional path to an image in the File Manager. See Common Conventions elsewhere in this document for more details.

ADD BUTTON
Displays the button for the user to click to add the Coupon to the order.

```
<[echo('shopping_cart','coupon','add_button','path to file')]>
```

Where `path to file` is an optional path to an image in the File Manager. See Common Conventions elsewhere in this document for more details.

COUPON FIELD

```
<[echo('shopping_cart','coupon','coupon_field')]>
```

VARIABLE: coupon_set
Gives you the ability to use Conditions to test whether or not a Coupon has been set, and is valid.

```
Values:  true / false
```

E-Commerce - Payment Form

Customizing the display of the payment page help maintain the look and feel of your site.

Elements

EDIT ADDRESS LINK
Returns the entire HREF to go to the Address Edit page.

```
<[echo('payment','address_link')]>
```

BILLING ADDRESS
Returns the shopper's HTML formatted billing address, based on the style settings in the store payment container.

```
<[echo('payment','billing_info')]>
```

SHIPPING ADDRESS
Returns the shopper's HTML formatted shipping address, based on the style settings in the store payment container.

```
<[echo('payment','shipping_info')]>
```

SHIPPING FORM ELEMENT
Provides the input for the shopper to change the current shipping option (if available).

```
<[echo('payment','shipping_form')]>
```

E-Commerce - Receipt Page

This code allows you to customize the display of the store receipt page.

Elements

ORDER NUMBER
Returns the order number of the just placed order.

```
<[echo('receipt','order_number')]>
```

BILLING ADDRESS
Returns the shopper's HTML formatted billing address, based on the style settings in the store payment container.

```
<[echo('receipt','billing_info')]>
```

SHIPPING ADDRESS
Returns the shopper's HTML formatted shipping address, based on the style settings in the store payment container.

```
<[echo('receipt','shipping_info')]>
```

CARD NUMBER
Returns the last 4 digits of the shopper's card number.

```
<[echo('receipt','card','number')]>
```

CARD EXPIRATION
Returns the shopper's card expiration date.

```
<[echo('receipt','card','exp')]>
```

COUPON CODE
Returns the coupon code used for the order.

```
<[echo('receipt','coupon_code')]>
```

Dynamic Data Display

There are several HyperCode values available inside of the dynamic data display container. They can be used to show a number next to each repetition, a total number of items in the list, etc.

FORMULA VALUE

```
<[echo_data(type="variable" class="record" name="formula name" stack="stack")]>
```

REPETITION COUNT:

```
<[echo_data(type="repeater" field="iteration")]>
```

REMAINING REPETITIONS:

```
<[echo_data(type="repeater" field="remaining")]>
```

REPETITION COUNT

The total number of items in the repeater:

```
<[echo_data(type="repeater" field="count")]>
```

Custom Login Form Display

To use the HyperCode below, you'll need to check the Customize Display box in the Login Container setup window. Once that is set, the Login Container will be treated just as any other container from a layout perspective. All of the calls below return basic form elements that reply on the surrounding from created by the login container.

Fields

PASSWORD
```
<[echo_data(type="login" field="password" size="chars" style="css")]>
```

USER ID
```
<[echo_data(type="login" field="handle" size="chars" style="css")]>
```

SUBMIT
```
<[echo_data(type="login" field="submit" image="path to file" style="css")]>
```

The optional field size takes a number to specify the width of the input element. You can include CSS tags in the optional `style` field. The submit button's optional image field take the path to a file in the File Manager. See Common Conventions elsewhere in this document for more details on using paths and the File Manager.

User Information

HANDLE/USER ID
```
<[echo('user_name','1')]>
```

FIRST AND LAST NAME
```
<[echo('user_name','2')]>
```

HANDLE WITH FIRST/LAST NAME
Look like: handle (first last)
```
<[echo('user_name','3')]>
```

EMAIL ADDRESS
```
<[echo('user_name','4')]>
```

USER KEY
```
<[echo('user_name','5')]>
```

FIRST NAME
```
<[echo('user_name','6')]>
```

LAST NAME
```
<[echo('user_name','7')]>
```

Current Date

This code is really handy for copyright dates. They'll change automatically when the year changes.

OUTPUT: January 20, 2008
```
<[echo('date','1')]>
```

OUTPUT: 1-20-08
```
<[echo('date','2')]>
```

OUTPUT: 08
```
<[echo('date','3')]>
```

OUTPUT: 2008
```
<[echo('date','4')]>
```

File Manager

HyperSites provides a File Manager to which you can upload your own files. You can find it in Toolbar > Site Tools > File Manager. Since there are no files stored on our application servers, your files are stored on any number of different media servers. To reference the files you've uploaded, use the following HyperCode.

ECHO FILE PATH

```
<[echo_data(type="file" path="path to file" server="local")]>
```

The `server` element will force our rendering system to load the file from our local media server. This ensures that the media's domain will always be media.hypersites.com so you can reference the files and patha in script. See Common Conventions elsewhere in this document for more details on using paths and the File Manager.

Session

If you're using the HyperSites User Actions API, you'll need to be able to reference the current session ID. For security reasons, we've encrypted the session ID, and provide access only to the encrypted value.

ENCRYPTED SESSION ID

```
<[echo_data(type="session" element="esid")]>
```

USER IP ADDRESS

```
<[echo_data(type="session" element="user_ip")]>
```

User Location

HyperSites is able to roughly determine your visitor's location using their IP address. It's a rough guess for many reasons, but it's about 80% accurate. You can use this data to simply display it back to the user, or you can use it as a filter in a Dynamic Data Display.

To enable the HyperSites location features, open: Toolbar > Site Tools > Site Setup > Advanced > Enable Location Services.

COUNTRY CODE
```
<[echo_data(type="user_location" element="country_code")]>
```

COUNTRY CODE 3 LETTERS
```
<[echo_data(type="user_location" element="country_code3")]>
```

COUNTRY NAME
```
<[echo_data(type="user_location" element="country_name")]>
```

REGION
```
<[echo_data(type="user_location" element="region")]>
```

CITY
```
<[echo_data(type="user_location" element="city")]>
```

ZIP
```
<[echo_data(type="user_location" element="zip")]>
```

LATITUDE
```
<[echo_data(type="user_location" element="latitude")]>
```

LONGITUDE
```
<[echo_data(type="user_location" element="longitude")]>
```

METRO CODE
```
<[echo_data(type="user_location" element="metro_code")]>
```

AREA CODE
```
<[echo_data(type="user_location" element="area_code")]>
```

COUNTY
```
<[echo_data(type="user_location" element="county")]>
```

USER IP ADDRESS
```
<[echo_data(type="user_location" element="user_ip")]>
```