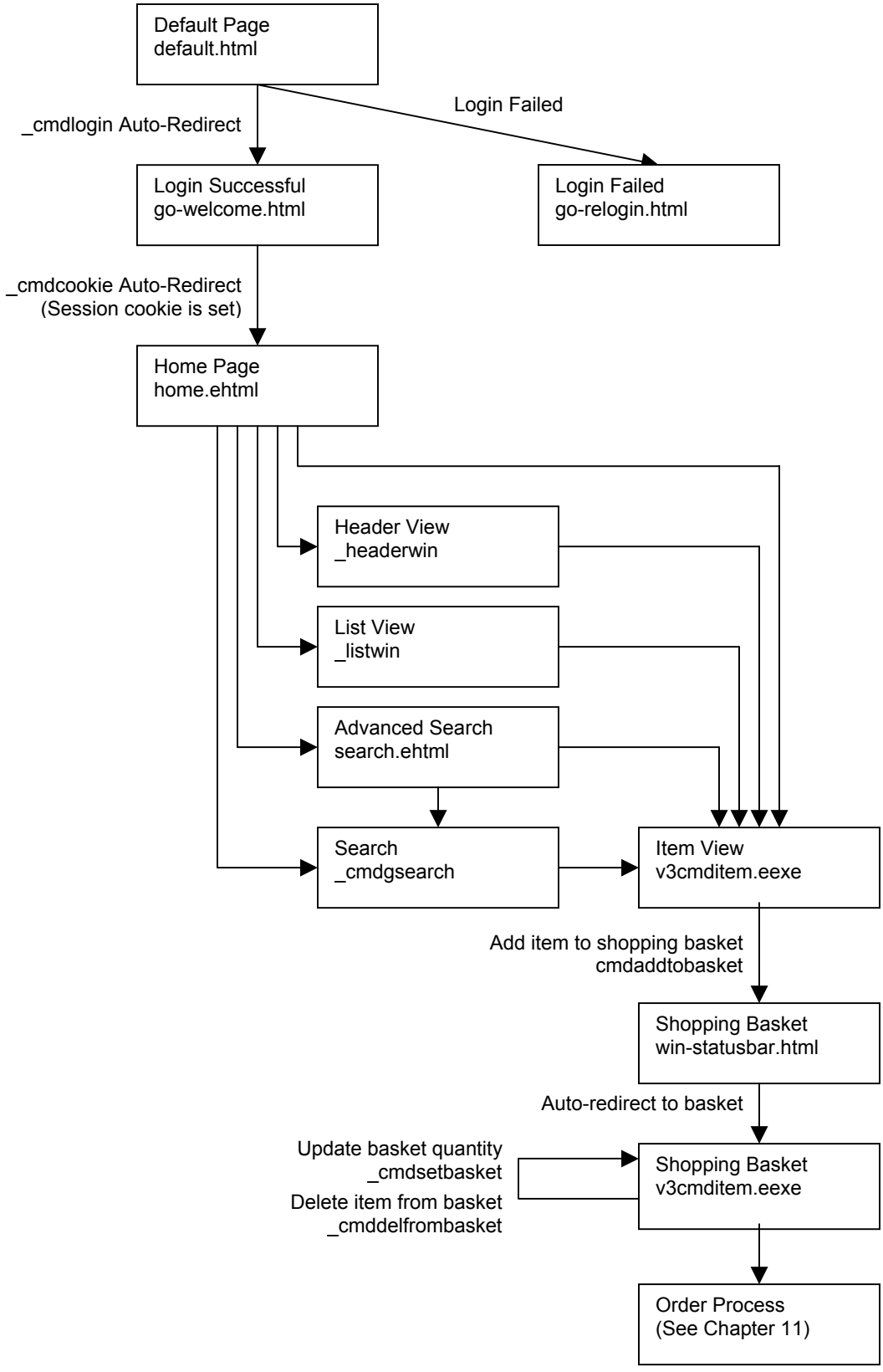
Appendix A
Tag and Command Reference

iNETstore Store Structure

An iNETstore shop is basically a set of web pages that retrieve and display information from a database, these pages are mostly made up of standard html code. The online store functionality is achieved through the use of iNETstore commands and tags.

The flowchart on the next page illustrates how stores built with iNETstore work.


Default Page
default.html
_cmdlogin Auto-Redirect
Login Failed
Login Successful
go-welcome.html
Login Failed
go-relogin.html
_cmdcookie Auto-Redirect
(Session cookie is set)
Home Page
home.ehtml
Header View
_headerwin
List View
_listwin
Advanced Search
search.ehtml
Search
_cmdgsearch
Item View
v3cmditem.eexe
Add item to shopping basket
cmdaddtobasket
Shopping Basket
win-statusbar.html
Auto-redirect to basket
Update basket quantity
_cmdsetbasket
Delete item from basket
_cmddelfrombasket
Shopping Basket
v3cmditem.eexe
Order Process
(See Chapter 11)

iNETstore Tags and Commands

Introduction

iNETstore uses special tags to extract information from the database and format it for display via a web browser. These tags operate in a similar manner to the HTML tags that web browsers interpret. The iNETstore tags are interpreted by the iNETstore Server and replaced with data from the database and inserted into the template as standard HTML which any web browser can understand.

Information Tags

The following tags apply to templates as indicated:

Tag	[DBxSUBSTITUTE]
Used in	win-searchresults.html win-header.html win-list.html win-registerresults.html win-reminderresults.html
Commands	_cmdgsearch _headerwin _listwin
The [DBxSUBSTITUTE] tag is used to print certain information that was previously written to the buffer by other tags in the files listed above.	

Tag	[DBxFIELD_fieldname]
Used in	All ehtml pages and most templates
This tag substitutes the content of the field with the name ‘fieldname’ from the applicable table. It is used in circumstances where the table name is already known by a previously processed function.	

Tag	[DBxFIELD_tablename_fieldname]
Used in	All ehtml pages and most templates

Substitutes the contents of the field with Field Name "fieldname" from the table "tablename".

Table name can have any of the following values:
ITM = Item table
CTG = Category table
FRT = Freight table
SYS = System table
USR = User table
TRN = Transaction table

This tag can be used in any ehtml page. For example, the tag [DBxFIELD_SYS_name] would be substituted with the content of a field called 'name' in row 0 of the system table.

Tag	[DBxARGS_argument]
Used in	All ehtml pages and most templates

This command is used to pass arguments between pages.

For example, you can pass an argument to an ehtml page as follows:

```
<a href="samplepage.ehtml?id=3">
```

If the tag [DBxARGS_id] is contained within samplepage.ehtml, it will be substituted by the argument that has been passed to the page. In this case, it would be substituted with '3'.

Tag	[DBxDBNAME]
Used in	All ehtml pages and most templates
Substitutes the name of the store.	

Tag	[DBxUSERNAME]
Used in	All ehtml pages and most templates
Substitutes the name of the currently logged in user.	

Tag	[DBxUSERGROUP]
Used in	All ehtml pages and most templates
Substitutes the group to which the user who is currently logged in belongs to.	

Tag	[DBxCLIENTIP]
Used in	All ehtml pages and most templates
Substitutes the IP address of the currently logged in user.	

Tag	[DBxTRANSACTIONID]
Used in	All ehtml pages and most templates
Substitutes the id of the transaction.	

Tag	[DBxNUMITEMS]
Used in	basket.ehtml
This tag is used between the [DBxBEGIN_CHILD_ITEM] and [DBxEND_CHILD_ITEM] tags. This tag is used to substitute the total quantity of the current item referenced in between these tags.	

Tag	[DBxTOTALOTHERSUM(x)]
Used in	basket.ehtml
Substitutes the total for each othersum(x) where x = 0...10 Example: If the _cmdaddtobasket function is used to add the item weights as othersum(1) then [DBxTOTALOTHERSUM(1)] will return the total weight in the cart.	

Tag	[DBxCURRENTDATE]
Used in	All ehtml pages and most templates
Substitutes the current date on the server.	

Tag	[DBxCURRENTTIME]
Used in	All ehtml pages and most templates
Substitutes the current time on the server.	

Tag	[DBxCURRENTDATETIME]
Used in	All ehtml pages and most templates
Substitutes the current date and time on the server.	

Template Specific Tags

Tag	[DBxBEGIN_BRANCH_PLUS] [DBxEND_BRANCH_PLUS]
Used in	Not used in the current templates
Commands	_headerwin _itemwin

These tags are used together to define the HTML code required to expand the navigation tree and may be placed anywhere in the page.

Example:

```
[DBxBEGIN_BRANCH_PLUS]
<a href="_headerwin?branchlevel=1">Expand Branch</a>
[DBxEND_BRANCH_PLUS]
```

Tag	[DBxBEGIN_BRANCH_MINUS] [DBxEND_BRANCH_MINUS]
Used in	Not used in the current templates
Commands	_headerwin _itemwin

These tags are used together to define the HTML code required to collapse the navigation tree and may be placed anywhere in the page.

Example:

```
[DBxBEGIN_BRANCH_MINUS]
<a href="_headerwin?branchlevel=0">Collapse Branch</a>
[DBxEND_BRANCH_MINUS]
```

Tag	[DBxBRANCHMODE]
Used in	Not used in the current templates
Commands	_headerwin?id=X _itemwin?id=X

This tag can be used in conjunction with the BRANCH tags above to provide end user feedback as to the branch mode the user is in.

Example:

```
[DBxBEGIN_BRANCH_PLUS]
<p><a href="_headerwin?branchlevel=1">Expand Branch</a><br>
      You are currently in: [DBxBRANCHMODE]</p>
[DBxEND_BRANCH_PLUS]
```

Tag	[DBxPAGES]
Used in	win-header.html win-list.html win-reminderresults.html win-searchresults.html
Commands	_headerwin _itemwin _cmdgsearch

This tag is used when the number of items to be displayed on a page exceeds the "maxpages" and "itemsperpage" defined during login or as defined in a search page using the _cmdgsearch command.

Example:

```
<p align="center">[DBxPAGES]</p>
```

If the number of items to be displayed requires 5 pages, then the following will be displayed where the tag above is located:

prev 1 2 3 4 5 next

Tag	[DBxBEGIN_SEARCH_CATEGORY] [DBxEND_SEARCH_CATEGORY]
Used in	Not used in the current set of templates
Commands	_cmdgsearch

These tags are used in conjunction with the tag [DBxFIELD_LISTBY] to sort search results by the field specified in the "listby" argument in the search page. These tags are useful if you would like to sort the search results by a field other than the "name" field such as "categoryid".

Example:

```
[DBxBEGIN_SEARCH_CATEGORY]
<p>[DBxFIELD_LISTBY]</p>
[DBxEND_SEARCH_CATEGORY]
```

Tag	[DBxBEGIN_CURRENT_CATEGORY] [DBxEND_CURRENT_CATEGORY]
Used in	win-header.html win-item.html
Commands	_headerwin _itemwin

Used to display the current category when in browse mode.

Example:

```
[DBxBEGIN_CURRENT_CATEGORY]
<p>Current category: [DBxFIELD_name]</p>
[DBxEND_CURRENT_CATEGORY]
```

Tag	[DBxBEGIN_PARENT_CATEGORY] [DBxEND_PARENT_CATEGORY]
Used in	win-header.html win-item.html
Commands	_headerwin _itemwin

Used to navigate back up to a parent category whilst in browse mode.

Example:

```
[DBxBEGIN_PARENT_CATEGORY]
<p><a href="_headerwin?id=[DBxFIELD_id]">Back to category:
[DBxFIELD_name]</p>
[DBxEND_PARENT_CATEGORY]
```

Tag	[DBxBEGIN_CHILD_CATEGORY] [DBxEND_CHILD_CATEGORY]
Used in	win-header.html win-item.html
Commands	_headerwin?id=X _itemwin?id=X

Used to display the categories that belong to the current category. Normally each of these categories has a hyperlink associated to them, which will hyperlink to the contents of that category.

Example:

```
[DBxBEGIN_CHILD_CATEGORY]
<a  href="_headerwin?id=[DBxFIELD_id]">[DBxFIELD_name]</a>
[DBxEND_CHILD_CATEGORY]
```

Tag	[DBxBEGIN_CHILD_ITEM] [DBxEND_CHILD_ITEM]
Used in	win-header.html win-list.html
Commands	_headerwin _itemwin _cmdgsearch (not used in current templates)

Used to display the items or products that belong to either:

- parent category (browse mode, i.e. from _headerwin or _listwin)
- search results (from a _cmdgsearch)

Example for win-header.html, win-item.html, win-searchresults.html:

```
[DBxBEGIN_CHILD_ITEM]
<a href="_itemwin?id=[DBxFIELD_ITM_id]">[DBxFIELD_ITM_name]</a>
[DBxEND_CHILD_ITEM]
```

Tag	[DBxBEGIN_NODATA] [DBxEND_NODATA]
Used in	win-header.html win-list.html win-registerresults.html win-reminderresults.html win-searchresults.html
Commands	_headerwin _listwin _cmdgsearch

These tags are used to insert a message to be returned when no data has been found in the browse or search mode.

Example:

```
[DBxBEGIN_NODATA]
<p>No data found.</p>
[DBxEND_NODATA]
```

Tag	[DBxBEGIN_SQL "sql-query"] [DBxEND_SQL]
Used in	All ehtml files and most templates

This set of tags is used to execute an SQL query.

Example:

```
[DBxBEGIN_SQL "SELECT name, id FROM ITM[DBxDBNAME] WHERE special LIKE
'yes' ORDER BY name"]
<a href="_v3cmditem.eexe?id=[DBxSQL_id]">[DBxSQL_name]</a><br>
[DBxEND_SQL]
```

The code in the example above will query the database for all items that are currently on special.

Note that there is a second set of SQL tags:
[DBxBEGIN_PURGEIFNORESULT_SQL "SQL"] and [DBxEND_SQL].

These tags can be used instead of [DBxBEGIN_SQL]. The difference is that if no records are found, anything between these tags is omitted.

Note that standard SQL tags and SQL tags with PURGEIFNORESULT tags can NOT be used on the same page. Otherwise a system error occurs.

As most templates use the [DBxBEGIN_PURGEIFNORESULT_SQL “SQL”] method it is recommended that this form of SQL statement be used.

More information about SQL is contained in one of the following sub-chapters.

Add To Basket Commands

Command	_cmdaddtobasket
Used in	All ehtml files and most templates

This command is used to add an item to the basket or update the quantity.

Parameters:

id	The id of the product, as specified in the item table.
dbname	The name of the store. Normally, the [DBxDBNAME] tag is used to populate this automatically. E.g. <input type="hidden" name="dbname" value="[DBxDBNAME]">
returnbasket	This value must be set to '0'.
othersum(x)	A sum to be added up. x can have any value between 0 and 9. By default, the following values are used: othersum(0) = Price inc tax othersum(1) = Price ex tax othersum(2) = Weight
varxxxxx	This parameter can be used for configurations such as colour, size, etc. Replace xxxxx with the name of the configuration, e.g. varcolour, varsize, etc.
numitems	Number of items to be added to the basket.
redirectlocation	This argument specifies the action to take after the command has been executed. Check out the section about the win-statusbar.html page for more information.

Example:

```
<form action="_cmdaddtobasket">
<input type="hidden" name="dbname" value="[DBxDBNAME]">
<input type="hidden" name="returnbasket" value="1">
<input type="hidden" name="othersum(0)" value="[DBxFIELD_priceinc]">
<input type="hidden" name="othersum(1)" value="[DBxFIELD_priceex]">
<input type="hidden" name="othersum(2)" value="[DBxFIELD_weight]">
<select name="varsize">
  <option value="small">
  <option value="medium">
  <option value="large">
</select>
<input type="text" name="numitems" value="" size="5">
<input type="submit" value=" Add To Basket ">
</form>
```

Tag	[DBxCONFIG_configname]
Used in	win-item.html
	This tag is used to display configurations, such as colour, size, etc. 'configname' represents the name of the field out of which configurations are drawn. In the database, configurations need to be delimited by a semi-colon. For example, as follows: **red;green;blue;black;white** Example: **<p>[DBxCONFIG_config1]<p>** If the field 'config1' contained 'red;green;blue', then the following would be displayed in the store: **<select size='1' name='varconfig1' class='dropdown'> <option>red <option>orange <option>blue </select>** Note that the class name of the resulting element is always 'dropdown'. This cannot be changed. Your cascading style sheet should refer to this.

Command	_storebasket
Used in	storecartwarning.ehtml
Template	result-storebasket.html

This command is used to store the content of an existing shopping basket in the database.

The command can only be used by registered customers who are logged into the store with their correct user name and password. It doesn't work with guest accounts.

In order for the command to work, the following two memo fields need to be present in the user table:

- storedbasket
- storedothertotal

Only one basket can be stored per account.

The command doesn't use parameters.

Command	_loadbasket
Used in	loadcartwarning.ehtml
Template	result-loadbasket.html

This command is used to retrieve the content of a previously stored shopping basket from the database.

The command can only be used by registered customers who are logged into the store with their correct user name and password. It doesn't work with guest accounts.

In order for the command to work, the following two memo fields need to be present in the user table:

- storedbasket
- storedothertotal

Only one basket can be stored per account.

The command doesn't use parameters.

Order Commands

Command	_v3cmdorder.eexe
Used in	checkout-NoFreight.ehtml checkout-FlatRate.ehtml checkout-TotalItems.ehtml checkout-TotalValue.ehtml checkout-Weight.ehtml
Template	No template

This command takes an order and redirects to the iNETstore Payment Gateway.

merchantemail	Email address of the merchant. Order notification is sent to this address.
merchantemailtemplate	File name of the template used to format the order notification email.
emailsubject	Subject line of the order notification email.
encfield(x)	Name of a field that is to be included in the encrypted string passed on to the iNETstore Payment Gateway. x must be a value between 0 and 9.
other parameters	Any other parameters that are supplied are written to the database. E.g. the content of a field called 'name' is written to the field called 'name' in the transaction table.

Note that all parameters that are passed on to the iNETstore Payment Gateway are contained in an encrypted string. Relevant details are contained in these fields in the system table:

key	The key used to encrypt the parameters that are passed on to the gateway.
gatewayaccountname	The gateway account name of the merchant.
gatewayurl	The URL of the iNETstore Payment Gateway. Note that this must start with 'https://'.

Tag	[DBxENCARGS_argumentname]
Used in	orderconfirmation.ehtml

This tag is used in the order confirmation page to decrypt an encrypted string that has been passed back from the iNETstore Payment Gateway.

Arguments that are passed back by default are:

transactionid	The transaction id of the order.
pwString	A random password string that has been generated by the _v3cmdorder.eexe command when the order was stored in the transaction table.

Note that the names of these arguments are case sensitive.

The arguments can be used to query the transaction table for order details to build an order confirmation page. View the orderconfirmation.ehtml file for an example of how this works.

Category and Item Page Commands

Tag	_headerwin
Template	win-header.html
Used in	home.ehtml In the browse box on most pages

Displays categories and items in the Header browse mode.

Argument	Mandatory	Input Type	Purpose
id	optional	optional	The id of the category to view (ID Field in CTG table).
branchlevel	optional	optional	To what extend the categories should be displayed together with their sub-categories or the items belonging to the category. The higher the number the more the branch will be expanded displaying the sub-categories and items underneath. Set to 0 - tree is collapsed Set to 1 - tree is expanded one category deep
itemorderby	optional	optional	The field in the ITM table which should determine the sort order of items.
ctgorderby	optional	optional	The field in the CTG table which should determine the sort order of categories.

Notes:

The header browse mode performs exactly the same function as the list browse mode. Having two browse modes enables maximum design flexibility by being able to specify two different browse modes for every category if necessary.

The arguments "branchlevel", "itemorderby" and "ctgorderby" can also be used as a global setting under the login command. If these arguments are also used for an individual category then the global setting will be over-ridden for that category.

Example:

```
<a href="_headerwin?id=[DBxFIELD_id]&branchlevel=1">
[DBxFIELD_name]
</a>
```

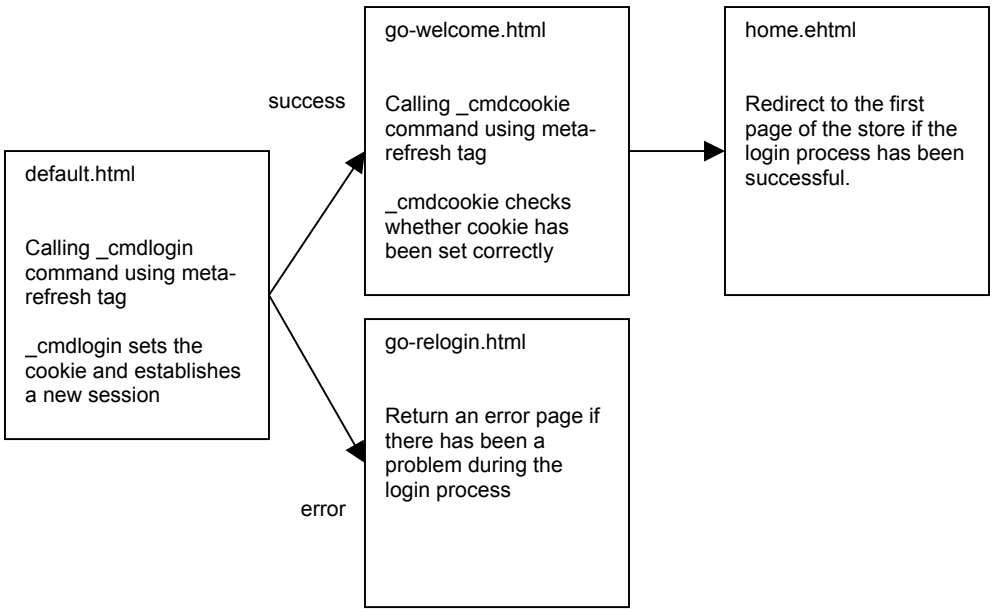
Tag	_v3cmditem.eexe
Used in	home.ehtml win-list.html win-header.html win-searchresults.html
Template	win-item.html

This command is used to display an item page. The only argument that is used with this command is ‘id’. Id should be the id of the item.

Example:

```
<a href="_v3cmditem.eexe?id=[DBxFIELD_id]">[DBxFIELD_name]</a>
```

Login Commands



When customers arrive at your store they must be logged in so that their product selections and orders can be processed correctly. The following files handle the login process:

- default.html
  This is the first page that the customer hits when they enter your store, generally it only consists of a Meta Refresh statement with the relevant login details.

- go-welcome.html
  This is the next page after default that the customer receives, once again it is a page with a immediate Meta refresh so is only displayed for a split second. The purpose of this page is to set a cookie on the customers computer to make tracking of the customers product selection and ordering easier.

- go-relogin.html
  This is the page that a customer will receive if there is any problem with logging into the store or if something goes wrong with the login during their visit to your iNETstore. Basically it redirects them back to the entry to your store to allow them to login again with fresh session details.

- home.ehtml
  This is the first page of the store. The location of this page is contained in the 'framemain' field of the user table in the database. In the case of the templates that ship with iNETstore, this page is called 'home.ehtml'.

Tag	_cmdlogin
Used in	home.ehtml and other pages that use a login box
Commands	n/a

The login command sets the parameters to be used to display information to a user as they browse through a store. The login command will execute a sequence of events as indicated in the schematic of the templates.

Arguments	Mandatory	Input Type "hidden"	Purpose
name	optional	optional	Name of user.
viewmode	optional	optional	Whether the LIST (win-list.html) or HEADER (win-header.html) view modes will be used to display information after the login.
branchlevel	optional	optional	To what extend the categories should be displayed together with their sub-categories or the items belonging to the category. The higher the number the more the branch will be expanded displaying the sub-categories and items underneath. Set to 0 - tree is collapsed Set to 1 - tree is expanded one category deep
itmsperpage	optional	optional	The number of items per page to be displayed. After this number of items have been displayed the tag [DBxPAGES] will hyperlink the user to another page with the remainder of the items.
maxpages	optional	optional	Controls the number of pages returned by the [DBxPAGES] tag.
categoryid	optional	optional	Specifies a default category the user should view when they login.
framemain	optional	optional	The file name of the first page to display after login.
itmorderby	optional	optional	The field in the ITM table which should determine the sort order of items.

Arguments	Mandatory	Input Type "hidden"	Purpose
ctgorderby	optional	optional	The field in the CTG table which should determine the sort order of categories.
login	yes	optional	The user login (Name field in USR table) required to login (Premium & Enterprise Ed. only).
password	yes, if user has a password	optional	The user password required to login (Premium & Enterprise Ed. only).
group	optional	optional	The name of the group the user belong to (Premium & Enterprise Ed. only).
version	yes	yes	Enterprise Editions merchants must specify the value of "enterprise" for the "version" argument.

Notes:

- Most templates use a page (default.html) that automatically executes this command by use of a META Refresh tag. The META Refresh tag makes a page automatically refresh and execute a URL containing the login command.
- The login command can also be used to hyperlink to a store from external servers such as a virtual server. In these circumstances the hyperlink should contain the full path of the store URL and the login command, for example: http://www.acme.com/_cmdlogin
- When using Enterprise Edition all parameters are defined in the USR table. It is only necessary to enter the arguments "login", "password" and "version". The remaining arguments are all defined in the USR table for the relevant user.

Example:

```
<META HTTP-EQUIV="refresh" CONTENT="0;
URL=_cmdlogin?name=guest&viewmode=HEADER&
branchlevel=0&itemsperpage=30&maxpages=5&framemain=home.shtml">
```

Command	_cmddologin
Used in	basket.ehtml
Template	result-dologin.html

This command resets the session to a different user name after a user has already been logged into the store.

For example, a user can be logged into the store as 'guest' using the normal login command. Using the _cmddologin command, the user can then be re-logged into the store with their normal login name and password.

Note: The difference between _cmdlogin and _cmddologin is that with _cmdlogin, a new session is established and the shopping basket content is lost. With _cmddologin, the existing session remains intact and the shopping cart content is preserved.

The template that is used by this command is result-dologin.html. This should contain two parts:

```
[DBxBEGIN_SUCCESS]
      <!-- This is executed if login has been successful -->
[DBxEND_SUCCESS]

[DBxBEGIN_FAIL]
      <!-- This is executed if login has failed -->
[DBxEND_FAIL]
```

Tag	_cmdcookie
Used in	go-welcome.html
Template	Redirects to the page specified in 'framemain' field
Commands	Used after _cmdlogin command has been executed.

This command places a cookie on the users system to track the session. If the client does not accept cookies, then the system will automatically use iNETstore's cookie-less session tracking technology (CSTT). CSTT includes a unique user id in the store URL.

When executed successfully, the command redirects the user to the page specified in the 'framemain' field of the record in the user table that corresponds with the user who is logged in.

The _cmdcookie command needs to be called as follows:

```
<META HTTP-EQUIV="refresh" CONTENT="0; URL=_cmdcookie?tb-
[DBxDBNAME]=[DBxSESSIONID]">
```

Search Function

Introduction

iNETstore allows you to create an advanced, fully customisable search function for your site. A search function normally consists of two html pages, one used to display the search form and one to show the search results.

The Search Page (search.ehtml)

This is a standard html page that contains a form through which customers can select search options and enter search terms.

File Name And Location

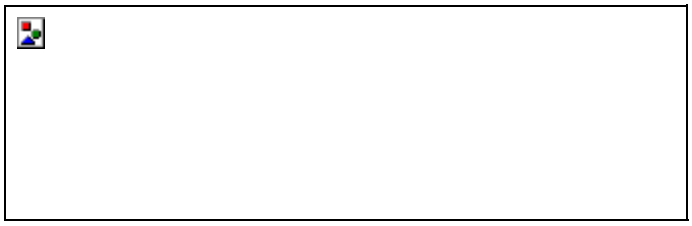
Most sample templates contain an advanced search page called 'search.ehtml'. Some templates also contain a simple search form in the header. Any page can contain a search form.

Search Form Structure

There are some fields that are mandatory in every search form. These are:

Name	Type	Optional or required	Description
sqlwhere	hidden	Required	The search string created by this function will be put into this field. This field must be present but should be empty.
sqlwhereOriginal	hidden	Required	Any additional search criteria specified by the developer.
searchfield	hidden	Required	Field or fields that are to be searched in the database. If multiple fields are specified, commas should delimit them. Please don't use spaces.
searchType	radio, select	Optional	Allows a user to select the search type they want to use. Valid types are "AND", "OR" and "PH" (see methods above). This element is only to be used if the "method" argument is either set to "RADIO" (for radio element) or "SELECT" (for select element).
search	text	Optional	The text field into which users can enter search terms.
dbtype	hidden	Required	Type of table that is searched, e.g.: ITM = Item table CTG = Category table FRT = Freight table USR = User table TRN = Transaction table
template	hidden	Required	Template that is use to format search results.
sqlorder	hidden	Required	Order in which search results are sorted. This field should contain a variable that will insert the value selected in the 'listby' field.
listby	hidden or select	Required	Order in which search results are sorted. If the user can select this, the field type should be 'select'. Otherwise, this can be a hidden field.

Search Form Example



```
<!-- When the form is submitted, the createSearch JavaScript function -->
<!-- is called. This is the function that enables Boolean searches -->
<form method="GET" action="_cmdgsearch" onSubmit="return createSearch(this,'RADIO')">

<!-- This is the template that is to be used to parse search results. -->
<input type="hidden" name="template" value="win-searchresults.html">
<!-- Database table that is to be searched. In this case, this is the item table. -->
<input type="hidden" name="dbtype" value="itm">
<!-- This field will be populated with SQL code by the createSearch JavaScript. -->
<!-- Any value in the sqlwhereOriginal field will be appended to allow you  -->
<!-- to specify additional search options. -->
<!-- If you don't use the createSearh script, you may enter your own SQL code -->
<!-- into this field (instead of the sqlwhereOriginal field). -->
<input type="hidden" name="sqlwhere" value="">
<!-- Enter SQL code for additional search options in this field. -->
<!-- The example below shows how to restrict your search to items that are -->
<!-- enabled. Searches are also restricted to items that fall within a certain -->
<!-- price range. Note how a variable is used to pick up the price range SQL -->
<!-- code from a pull-down menu. -->
<input type="hidden" name="sqlwhereOriginal" value="
{ enabledweb = 1 }
{ AND }
{ $priceRange$ }
">
<!-- A variable is used here to pick up the order in which items are to -->
<!-- appear in the search results from a pull-down menu. -->
<input type="hidden" name="sqlorder" value="{ $listby$ }">
<!-- The names of fields that are to be searched are specified here. -->
<!-- If you want to search multiple fields, delimit them with commas. -->
<!-- Note: It is advisable not to search memo fields since such searches -->
<!-- will negatively affect the performance of the database. -->
<input type="hidden" name="searchfield" value="name,keywords">

<!-- This is the box where people can enter search terms. -->
Search Text:
<input name="search" type="text" SIZE="30"><br>
<!-- People select the price range from the pull-down menu below.  -->
<!-- The user selected value is picked up by the iNETstore server -->
<!-- because of the variable used in the sqlwhereOriginal field. -->
Price range:
<select name="priceRange">
  <option value="priceInc < 50">Under $50
  <option value="priceInc >= 50">$50 or more
</select>

<!-- Users can select how they want to search. -->
Match:
<input type="radio" name="searchType" value="OR" checked>Any terms  
<input type="radio" name="searchType" value="AND">All terms  
<input type="radio" name="searchType" value="PH">Exact phrase<br>

<!-- The pull-down menu below gives users the option to decide how they want -->
<!-- to have the search results sorted. -->
List by:
<select name="listby">
<option value="name">Product name
<option value="priceInc">Price
<option value="categoryId">Category
</select><br>
<!-- This is the button on which people can click to start a search. -->
<input type="submit" value="Search">

</form>
```

Using Advanced Boolean Searches

iNETstore supports advanced Boolean database searches through the use of a JavaScript function. This function, called 'createSearch', is included in the scripts.js file that comes with all templates. To enable Boolean searching, the createSearch function must be called in the form tag as follows:

```
<form method="GET" action="_cmdgsearch" onSubmit="return createSearch(this,'RADIO')">
```

The function will parse user-specified search terms and create an SQL search string for use by the iNETstore server.
Following is a brief description of the function:

Function Name

createSearch(form [, method])

Arguments

form — This is the form object that contains the input fields required by the function.Normally, if the function is called as described above, this argument should be'this' (see example).

method — The method is optional. This is a string, which must be contained betweenquotes. The default value is AND. This means that if no method is specified, thesearch will search return all rows that contain all terms. A list of validmethods is shown below:

AND	Search results must match all specified terms.
OR	Search results should match any one or multiple specified search terms.
PH	Search results must match the entered phrase exactly.
RADIO	HTML radio elements are used to give users the choice between the above search methods.
SELECT	A HTML drop-down menu is used to give users the choice between the above search methods.

Return Value

The script automatically validates user input. If the validation is not successful (e.g. if the user did not specify any search terms), then the function returns false and submission of the form content to the iNETstore server should not go ahead.

True	User input validation has been successful.
False	User input validation has not been successful. This is caused by users not entering a valid search term. If the return value is False, the user will be prompted to enter a valid search term.

Boolean search examples

Below are a few examples that demonstrate how Boolean search functionality can be implemented with iNETstore.

Example 1 - Simple, Search Will Match Any Specified Term

Features of this search form:

- This is the most simple search form, returning results that match any of the specified search terms.

```
<form METHOD="GET" ACTION="_cmdgsearch"
onSubmit="return createSearch(this,'OR')">
<input type="hidden" name="template" value="win-searchresults.html">
<input type="hidden" name="dbtype" value="itm">
<input type="hidden" name="sqlwhere" value="">
<input type="hidden" name="sqlwhereOriginal" value="">
<input type="hidden" name="sqlorder" value="{ $listby$ }">
<input type="hidden" name="listby" value="name">
<input type="hidden" name="searchfield" value="name">
Search terms:
<input type="text" name="search" size="30">
<input type="submit" value=" Search ">
</form>
```

Example 2 - Advanced, Using Drop-Down Menu

Features of this search form:

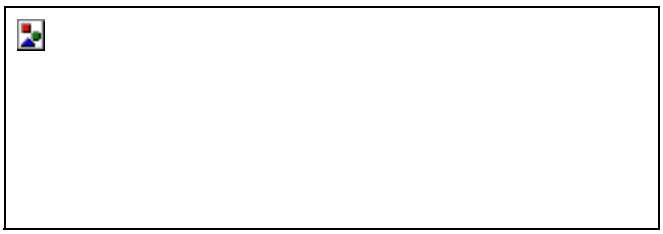
- Users can select the required search method from a pull-down menu
- Only the name field is searched

```
<form METHOD="GET" ACTION="_cmdgsearch"
onSubmit="return createSearch(this,'SELECT')">
<input type="hidden" name="template" value="win-searchresults.html">
<input type="hidden" name="dbtype" value="itm">
<input type="hidden" name="sqlwhere" value="">
<input type="hidden" name="sqlwhereOriginal" value="">
<input type="hidden" name="sqlorder" value="{ $listby$ }">
<input type="hidden" name="listby" value="name">
<input type="hidden" name="searchfield" value="name">
Search terms:
<input type="text" name="search" size="30">
<br>Match:
<select name="searchType">
  <option value="OR">Any Terms
  <option value="AND">All Terms
  <option value="PH">Exact Phrase
</select>
<input type="submit" value=" Search ">
</form>
```

Example 3 - Advanced, using radio check-box

Features of this search form:

- Users can select the required search method through radio check-boxes)
- Name, description and keyword fields are searched
- Additional search restraints are used (price range)



```
<form METHOD="GET" ACTION="_cmdgsearch"
<input type="hidden" name="sqlwhere" value="">
<input type="hidden" name="sqlwhereOriginal" value="
{ $priceRange$ }
{ AND }
{ enabled = 1 }
">
<input type="hidden" name="sqlorder" value="{ $listby$ }">
<input type="hidden" name="listby" value="name">
<input type="hidden" name="searchfield" value="name,description,keywords">\
Search terms:
<input type="text" name="search" size="30">
<br>Match:
<input type="radio" name="searchType" value="OR" checked>Any terms
<input type="radio" name="searchType" value="AND">All terms
<input type="radio" name="searchType" value="PH">Phrase
<br>Price range:
<select name="priceRange">
<option value="priceInc >= 0">Any price
<option value="priceInc <= 20">$20 or less
<option value="priceInc >= 20 AND priceInc <= 50">$20 to $50
<option value="priceInc >= 50 AND priceInc <= 100">$50 to $100
<option value="priceInc >= 100">$100 or more
</select>
<br>
<input type="submit" value=" Search ">
</form>
```

Building customised queries

Queries can be extended and customised with a language similar to SQL. Customised queries should be contained in a field called 'sqlwhereOriginal'.

Logical operators that are supported are AND and OR.

Allowed comparison operators are = (equal), <> (not equal), < (less), <= (less than or equal to), > (greater than) and >= (greater than or equal to).

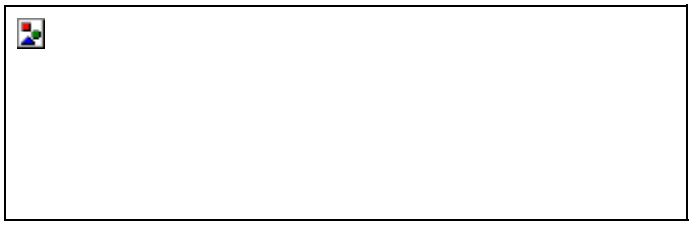
You can also use variables. Variable names must start and end with a dollar sign. During execution, the iNETstore server substitutes variables with the value of the form field with the same name. For example, $searchterms$ will be substituted with whatever a user enters into the search field (<input type="text" name="searchterms" value="">).

Each line of the query must be contained within brackets { }.
Percentage signs can be used to represent any possible characters. A few examples are shown below:

	Description	Matches	Doesn't match
%H	Any database entry that ends with the letter H will be matched.	Bill Smith	Bill Jones
B%	Any database entry that starts with the letter B will be matched.	Bill Smith	Robert Bell
%E%	Any database entry tat contains the letter E will be matched.	Robert Bell	Bill Smith

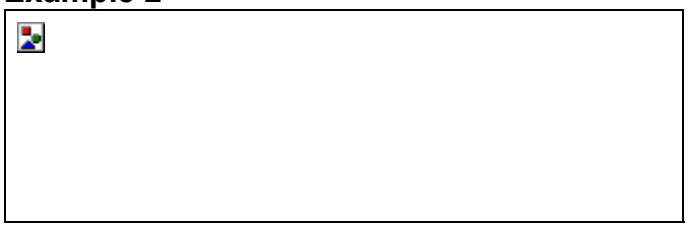
Note: By default, SQL is not case sensitive. Thus, it doesn't matter whether your specified search terms are in upper or lower case characters.

Example 1



```
<input type="hidden" name="sqlwhereOriginal" value="
  { enabledweb = 1 }
  { AND }
  { $priceRange$ }
">
<select name="priceRange">
  <option value="priceInc < 50">Under $50
  <option value="priceInc >= 50">$50 or more
</select>
```

Example 2



```
<input type="hidden" name="sqlwhereOriginal" value="
  { enabledweb = 1 }
  { AND }
  { ( }
  { priceInc >= $priceFrom$ }
  { AND }
  { priceInc <= $priceTo$ }
  { ) }
">

From:
<select name="priceFrom">
  <option value="0">$0.00
  <option value="20">$20.00
  <option value="50">$50.00
</select>

To:
<select name="priceTo">
  <option value="20">$20.00
  <option value="50">$50.00
  <option value="100">$100.00
</select>
```

Searches without using our JavaScript

If you don't require capability to do Boolean searches, you can create a search page that does not use the iNETstore createSearch JavaScript function. In this case, you can enter your customised SQL query directly into the hidden field called 'sqlwhere' and the following fields will not be required: sqlwhereOriginal', 'searchfield', 'searchType' and 'search'.

The search results page (win-search.html)

File name and location

The file name of the search results page is specified in the search form in a hidden field called 'template'. The default file name is 'win-searchresults.html'. This name is used in most of the sample templates that come with iNETstore.

The search result page is stored in the 'templates' folder. The correct path is as follows:
C:**iNETstorePath**\public-html\catalogue**storename**\templates**usergroup****win-searchresults.html**

where:

C = The letter of your drive
iNETstorePath = The path to the iNETstore installation
storename = The name of your store.
usergroup = The user group, e.g. 'public'.
win-searchresults.html = This can be any file name that is specified in the search form.

Code structure

iNETstore uses a simple structure to format search results. This is shown below:

iNETstore Code	Comments
`<html>`	

`[DBxBEGIN_NODATA]`
`No data was found`
`[DBxEND_NODATA]`
	If the search does not return any results, this code is stored in the memory cache of the iNETstore server.

`[DBxBEGIN_CHILD_ITEM]` `- [DBxFIELD_name] ` `[DBxEND_CHILD_ITEM]`	If the search finds records, iNETstore will loop through them here and write the result to the memory cache. In this case, only one field, called 'name' is shown. You can shown any field that is available in the table that is queried.

`<p>`
`[DBxSUBSTITUTE]`
`</p>`
	This is where the results are inserted into the search results field (from the memory cache).

`<p>[DBxPAGES]</p>`	If the search returns multiple pages, links to additional pages are shown here.
`</html>`	

Submitting Data Using _cmdsubmit

The _cmdsubmit command can be used to submit data to the database.

Parameters

id	This is the id of the record to be modified. If you wish to create a new record, set the id to -1 (minus one). Note that the id is mandatory. If no valid id is specified, no records will be modified.
ctgid	The id of the parent category to be modified. This is only required if you modify data in the item table (ITM). If you modify data in any other table, you should not use this argument.
parentid	The id of the parent category to be modified. This is only required if you modify data in the category table (CTG). If you modify data in any other table, you should not use this argument.
dbtype	The name of the table to be modified. For example, this can be ITM, CTG, TRN, SYS, etc.
dbname	The name of the store.
emailto	The email address to which the iNETstore Server will send notification of the database change.
xxx	The name of any field that is to be updated.

Note:
For security reasons, submit will fail if a record is submitted to the user table (USRstorename) and the parameter called 'usergroup' is 'admin'. This has been done to prevent unauthorised users from registering an admin user account.

Example

_cmdsubmit?id=-1&password=[DBxFIELD_SYS_password]&dbtype=ITM&
dbname=mystore&emailto=updates@inetstore.com.au

Note: Alternatively to using the _cmdsubmit command, data can be submitted to the database using a standard ‘UPDATE’ SQL statement. Check out the SQL section in the appendix for more information.

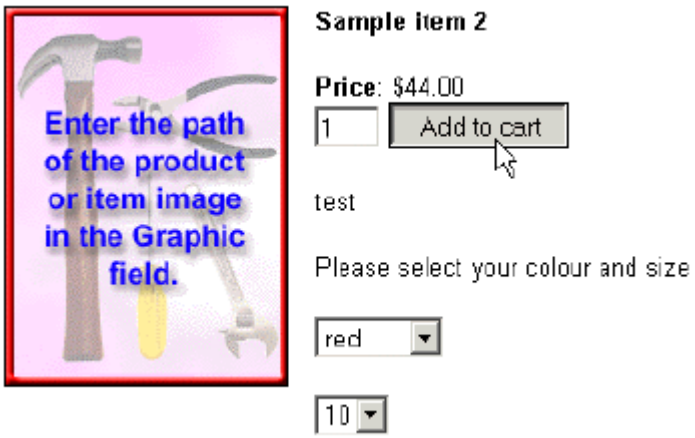
Shopping Basket

Basket Methods

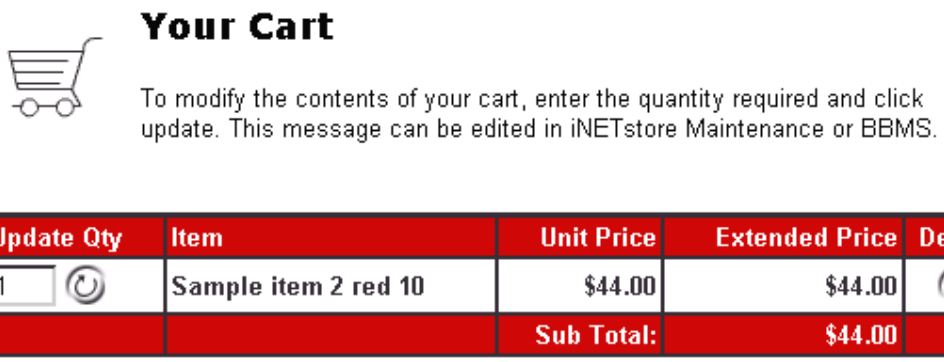
The default templates that ship with iNETstore support two different basket methods:

Method A

Customers are always shown the shopping basket page when they add an item. This is the default method that is suitable for most stores.

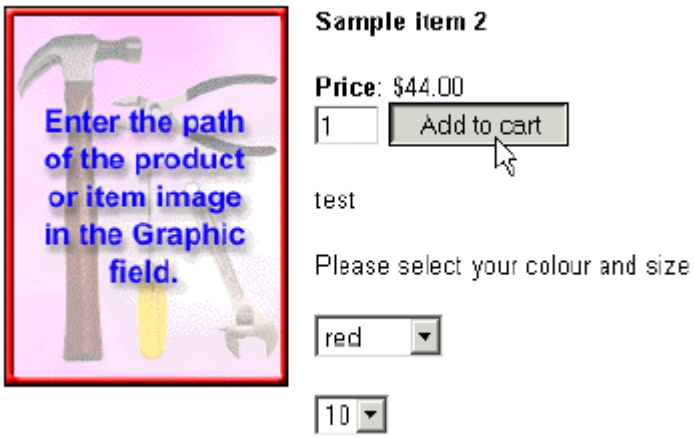


1. A user clicks on the 'Add to cart' button to add an item to the shopping basket.
2. The user is automatically redirected to the shopping basket page

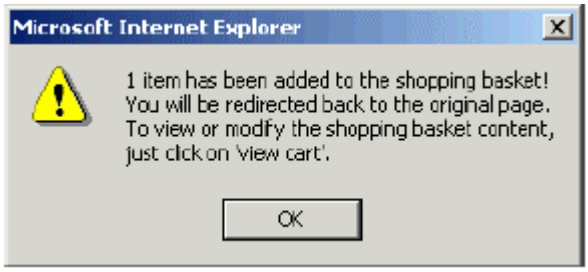


Method B

When customers add an item, they are shown a brief message that the item has been added. iNETstore then redirects back to the original page. For example, this might be useful in the case of a supermarket, where it is better to display many items on a single page. The customer can then add multiple items without always being redirected to the shopping basket page. The images below illustrate the process:



1. A customer clicks on the 'Add to cart' button to add an item to the shopping basket.



2. iNETstore then displays a short message saying that the item has been added to the basket.

3. Upon clicking on the 'OK' button, the user is redirected to the original page.

Changing The Basket Method

To change the basket method, follow these steps:

Changing From Method A to B

1. Open a file called 'win-statusbar.html'. This is located in the templates directory.

2. At the top of the file you can see the following line:
```
<META HTTP-EQUIV="refresh" CONTENT="0; URL=../[DBxUSERGROUP]/basket.ehtml">
```
This must be changed to:
```
<!--META HTTP-EQUIV="refresh" CONTENT="0; URL=../[DBxUSERGROUP]/basket.ehtml"-->
```

3. Further down, you can see this line:
```
<body class="body" onLoad="//redirectBrowser();">
```
Change it to:
```
<body class="body" onLoad="redirectBrowser();">
```

4. Done. iNETstore is now configured to use basket method B.
Note: If your store contains multiple user groups, you must carry out the above procedure with the win-statusbar.html file of each user group.

Changing From Method B to A

To change back from method B to A you will need to carry out the reverse procedure:

1. Open a file called 'win-statusbar.html'. This is located in the templates directory.

2. At the top of the file you can see the following line:
```
<!--META HTTP-EQUIV="refresh" CONTENT="0; URL=../[DBxUSERGROUP]/basket.ehtml"-->
```
This must be changed to:
```
<META HTTP-EQUIV="refresh" CONTENT="0; URL=../[DBxUSERGROUP]/basket.ehtml">
```

3. Further down, you can see this line:
```
<body class="body" onLoad="redirectBrowser();">
```
Change it to:
```
<body class="body" onLoad="//redirectBrowser();">
```

4. Done. iNETstore is now configured to use basket method A.
Note: If your store contains multiple user groups, you must carry out the above procedure with the win-statusbar.html file of each user group.

Using Method B

When using method B, you should ensure that your add to basket forms contain a hidden field named 'redirectlocation'. I.e. they must contain the following line:

```
<input type="hidden" name="redirectlocation" value="0">
```

The values are as follows:

0 This value should be used in an add to basket form, i.e. if somebody adds an item to the shopping basket. It causes the browser to return to the original page.

1 This value should be used in the form where customers update the shopping basket quantity. It causes the browser to refresh the basket after the new quantity has been submitted.

2 This value should be used in the form where customers delete items from the shopping basket. It causes the browser to refresh the basket after the new quantity has been submitted.

default If no value or another value is used, a brief message is shown, explaining to the user that the item has been added to the shopping basket. The browser is then redirected to the shopping basket page.

Stock Take

Displaying The Stock Level

Displaying Absolute Stock Levels

To display the stock level in your store, you can use the following tags:

[DBxFIELD_ITM_stocklevel] Display the number of items that are in stock.

[DBxFIELD_ITM_backorder] Display the number of items that are on backorder.

Displaying a Stock Level Range

Sometimes, it is better just to give an indication of the number of items that are available, without showing the exact number. You can use the JavaScript below to do that:

```
<script language="JavaScript">
<!--
document.write("Stock level: ");
if ([DBxFIELD_ITM_stocklevel] > 10) {
          document.write("more than 10 items");
} else {
          document.write("10 or less items");
}
// -->
</script>
```

If more than 10 items are in stock, the output of this script is:

Stock level: more than 10

If less than 10 items are in stock, the output is:

Stock level: 10 or less

Obviously, you can customise the script to suit your requirements.

Removing Items That Are Out Of Stock

It is possible to automatically remove items that are out of stock from your store. To achieve this, you will need to include a simple SQL statement in your order confirmation page (orderconfirmation.ehtml) page. The statement looks like this:

```
[DBxBEGIN_SQL "UPDATE ITM[DBxDBNAME] SET enabledweb = 0 WHERE stocklevel = 0"]
[DBxEND_SQL]
```

The statement will set 'enabledweb' fields of all items that are out of stock to 0 (zero). Items with this flag set to zero are not shown in the store.

Note that this method does not cause items to be deleted. They are only temporarily disabled and can be enabled again by setting the 'enabledweb' flag to 1 (one).

Editing Pages

One of the most useful features of iNETstore is the ability to fully customise the look and feel of your store. Changing the look of the store involves editing the pages that make up the store. There are three types of pages that make up a standard iNETstore, plain html files, dynamic ehtml files and iNETstore template files.

html files

Standard html files are used in your iNETstore for static pages, that is pages that contain no information that will vary from visit to visit. These files have a '.html' extension after their name and contain standard html code. Standard html pages are usually located in the root directory of your store or in the user group directories. Examples would be the default.html page and any other page with a standard set of information.

NOTE: iNETstore tags will not work in '.html' files.

ehtml files

Dynamic ehtml files are used throughout iNETstore where a page has changing content, such as information that is retrieved from the database or information that is dependant upon user input. These files have a '.ehtml' extension and contain mostly standard html code with the addition of some iNETstore tags and commands.

Dynamic ehtml files are found in the root directory of your store or in the user group directories. Examples would be the home.ehtml page and any other page that displays information that is contained in the database.

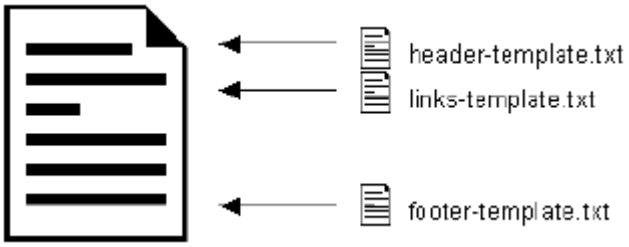
iNETstore template files

iNETstore template files are used to display information that is dynamically generated as a result of an iNETstore command. These files have a '.html' extension and contain a mixture of standard html code and iNETstore tags. iNETstore template files are found in the 'templates' directory of your store. Examples of iNETstore template files would be win-item.html and win searchresults.html.
Each iNETstore command has a default template file

Include Files

Overview

'Includes' are files that are included within another page when that page is parsed by the iNETstore Server. It is useful to put content that is part of many pages, such as page headers and footers, into include files. By using an include file content can be updated in a single file and reflected right throughout the web site.



The diagram illustrates how three files header-template.txt, links-template.txt and footer-template.txt are included in every page of an iNETstore shop.

Include Tag

The tag used to include files within a page looks as follows:

```
[DBxINCLUDE "filename.txt" FILE]
```

Replace filename.txt with the name of the file that is to be included. All include files should be stored in the relevant templates directory. For example, if you are working on the template for the public user group, you should store include files in this location:

c:\Program Files\iNETstore\public-html\catalogue\storename\templates\public

Replace 'storename' with the name of your store. If you are working on the template that belongs to another user group, you should replace 'public' with the name of that user group.

How To Create Your Own Include Files

To create your own include file, just follow these steps:

1. Create a new (text) file in the template directory of the relevant user group.
2. Edit the new file to add the required content.
3. Include the file in the original page using the DBxINCLUDE tag as described above.

With this method, files can be included in all ehtml and template files. However, the DBxINCLUDE tag does not work in standard html files because the iNETstore server does not parse them.

Securing EHTML Pages

With iNETstore, users are only allowed to access template files of the user group they belong to. However, by default users are allowed to access any .ehtml files. To prevent certain user groups from accessing particular .ehtml files, the DBxPERMIT tag can be used. This works as follows:

```
[DBxPERMIT(group1,group2,group3,...)]
```

The DBxPERMIT tag should be quoted out and used straight after the <html> tag at the top of the .ehtml page, as shown below:

```
<html>
<!--[DBxPERMIT(group1)]-->
<head><title>Sample Page</title></head>
<body>
Page content goes here ...
</body>
</html>
```

Example 1

```
<!--[DBxPERMIT(group1)]-->
```

Only the group called "group1" is allowed to access this .ehtml page. If a member of another group tries to access the page, an error message is shown.

Example 2

```
<!--[DBxPERMIT(group1,group3,reseller)]-->
```

All members of the groups called "group1", "group3" and "reseller" are allowed to access the page. If a member of another group tries to access the page, an error message is shown.

WARNING
If the DBxPERMIT tag is not used in a .ehtml page, all users (even public users) are allowed to access its content.