
OPEN SYSTEMS® Accounting Software

**OSAS® Web B2B
User's Manual**

2210.OW70

© 2005 Open Systems Holdings Corp. All rights reserved.

Document Number 2210.OW70

No part of this manual may be reproduced by any means without the written permission of Open Systems, Inc.

OPEN SYSTEMS and OSAS are registered trademarks and Resource Manager, Resource Manager for Windows, and Report Writer are trademarks of Open Systems Holdings Corp. BB^x is a trademark and PRO/5 and Visual PRO/5 are registered trademarks of BASIS International Ltd. Novell and NetWare are registered trademarks of Novell, Inc. Microsoft, Windows, Windows 95, Windows 98, Windows NT, Windows 2000 and MS-DOS are either trademarks or registered trademarks of Microsoft Corporation. Adobe and Acrobat are registered trademarks of Adobe Systems, Inc. TrueType is a registered trademark of Apple Computer, Inc.

Open Systems Accounting Software for UNIX uses PKZIP utilities from PKWARE, Inc.,
648 N. Plankinton Ave, Suite 220; Milwaukee, WI 53203. Phone: (414) 289-9788 Internet: www.pkware.com

Printed in U.S.A. This manual is recyclable.

November 2005, Release 7.0

This document has been prepared to conform to the current release version of OPEN SYSTEMS Accounting Software. Because of our extensive development efforts and our desire to further improve and enhance the software, inconsistencies may exist between the software and the documentation in some instances. Call your customer support representative if you encounter an inconsistency.

Open Systems, Inc.
4301 Dean Lakes Boulevard
Shakopee, Minnesota 55379

General Telephone	(952) 403-5700
General Fax	(952) 496-2495
Support Telephone	(800) 582-5000
Support Fax	(952) 403-5870
Internet Address	www.osas.com

Contents

Introduction

Welcome to OSAS	1-3
The OSAS Web B2B System	1-5
Starting OSAS	1-11
Navigating OSAS	1-15

Installation and Setup

Installing the Web Server	2-3
Installing OSAS Web B2B	2-5

File Maintenance

Item Group Codes	3-3
Inventory Item Pictures	3-5
Internet Inventory Item Groups	3-9
Customer Internet Access Codes	3-13
Customer Groups	3-17
Tables	3-21
Change Fields	3-23

Set Up Web Components

Create Login Page	4-3
Install Web Server Components	4-5

Sales Order Processing

Sales Order Processing on the Web	5-3
Transaction Journal	5-5
Build Sales Orders from Remote Files	5-7
Purge Log File	5-11

Remote Access

Remote Access	6-3
Copy Data Files to Web Server	6-5
Copy OSAS Programs to Web Server	6-7

Master File Lists

Printing a Master File List	7-3
Item Group Codes List	7-5
Inventory Item Pictures List	7-7
Internet Companies List	7-9
Internet Inventory Item Groups List	7-11
Internet Access Codes List	7-13
Customer Groups List	7-15
Tables List	7-17

The Web Interface

OSAS Web Login Page	8-3
Account Information	8-5
Aged Trial Balance	8-7
History Inquiry	8-9
Invoice Inquiry	8-11
Item Inquiry	8-13
Order Inquiry	8-19
Order Entry	8-21

References

System Messages	A-1
Common Questions	B-1
List of Files Copied	C-1
Editing Files	D-1
Files for Remote Access	E-1
Basic Web Utility	F-1
The Basic Web Utility and CGI	F-3
Data Templates	F-17
Sample Program Reference	F-21

Suggestions and Tips	F-37
Standard Structure Options	F-39
HTTP Cookies	F-41
Automatic Session Tracking	F-43
File Uploading Support	F-45
Global String Reference	F-47
Toolkit Program Reference Overview	F-53
Toolkit Program Functional Listing	F-57

Index	IX-1
--------------	-------------

Introduction

1

Welcome to OSAS	1-3
The OSAS Web B2B System	1-5
Starting OSAS	1-11
Navigating OSAS	1-15

Welcome to OSAS

Welcome to the OSAS Web B2B application for OPEN SYSTEMS Accounting Software® (OSAS®). OSAS Web B2B allows your customers to access selected information from your OSAS data files with any web browser. Your customers can also use the OSAS Web B2B system to enter sales orders into your system through the Internet.

OSAS Web B2B is different from other OSAS applications because of the web interface. It helps to think of the software as two pieces that are joined together—the web server application and the OSAS application.

OSAS Web B2B plugs into Resource Manager, the foundation of OSAS. Consult the Resource Manager guide for more information on basic OSAS functionality and details on how Resource Manager works within the OSAS system.

About This Guide

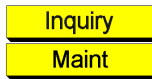
This guide describes the functions that make up the OSAS Web B2B application and gives details on how OSAS Web B2B fits into your existing business workflow. This guide is divided into these sections:

- Chapter 1 introduces OSAS and the OSAS Web B2B application, and describes the basics of the OSAS Web B2B system and how to navigate around OSAS.
- Chapter 2, Installation and Conversion, details how to install OSAS Web B2B using Resource Manager and how to create or convert the data files it requires.
- Chapter 3, Getting Started, gives information and checklists on the steps you need to perform to set up OSAS Web B2B.
- Chapters 4 through 12 contain function descriptions organized by menu. These chapters mirror the order that appears on the OSAS Web B2B menu.

- The Appendixes contain supplementary material not directly related to OSAS Web B2B functionality.
- The Index is a topical reference to the information in the rest of the chapters, and concludes this guide.

Conventions

This guide uses the following conventions to present information.



When the **Inquiry** or **Maintenance** commands (or both) are available for a field, the Inquiry and Maint flags appear in the margin. See page 1-22 and page 1-27 for more information on these commands.

When you see the phrase “use the **Proceed (OK)** command” in this guide, press **Page Down** in either text or graphical mode to continue. In graphical mode, you can also click **OK** to proceed.

The OSAS Web B2B System

B2B is a software industry term that describes web-based systems that facilitate business-to-business transactions. OSAS Web B2B is just such a system. You can use the OSAS Web B2B application to allow your customers access to selected information from your OSAS data files with any web browser. Your customers can also use the OSAS Web B2B system to enter sales orders into your system through the Internet.

You must have Internet access and Internet server software in addition to OSAS Web B2B. The Internet server software you use must support *Standard CGI* scripting in order to use it with OSAS Web B2B.

Note: CGI is an acronym for Common Gateway Interface, a method of communicating information from one web page to another. There are two types of CGI scripting standards: Windows CGI and Standard CGI. OSAS Web B2B uses Standard CGI.

A familiarity with *HTML* (Hypertext Markup Language) is recommended if you want to modify the web pages constructed by the **Create OSAS Web B2B Login Page** and **Install Web Server Components** functions.

OSAS Web B2B is different from other OSAS applications because of the web interface. It helps to think of the software as two pieces that are joined together—the web server application and the OSAS application.

OSAS Web B2B on the Web Server

The web server application is the part of OSAS Web B2B that your customers see. It is a connected group of HTML screens, some of which function as menus, and others that display and prompt for information. These screens communicate with OSAS Web B2B using standard CGI scripts.

Your Internet server software must be installed and configured on the web server before you can use OSAS Web B2B. The web server can be located on the same computer on which you run OSAS, or it can be located on a different computer. If you choose to use a different computer for a web server, you must be able to communicate between your OSAS system and the web server through a network or modem connection.

The web server and OSAS must be able to communicate because OSAS Web B2B provides several functions that move data files and programs from OSAS to the web server and back again.

If you run your web server on a different system than OSAS, and the two systems cannot communicate directly, you will need to move these files and programs manually using software designed for that purpose, such as FTP (File Transfer Protocol) software.

For details about the HTML screens used with OSAS Web B2B, see chapter 8.

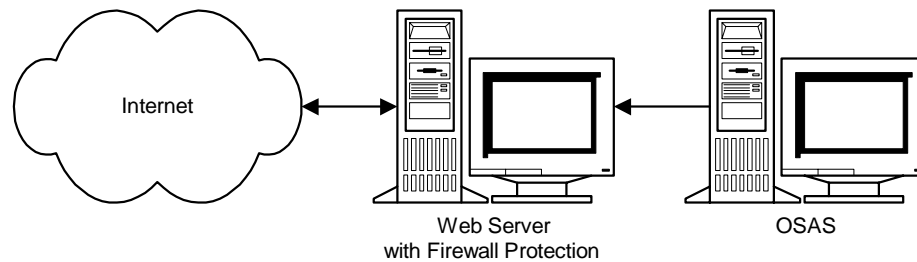
OSAS Web B2B Functions in OSAS

The OSAS side of OSAS Web B2B includes several functions that you use to set up and maintain the web server side, including:

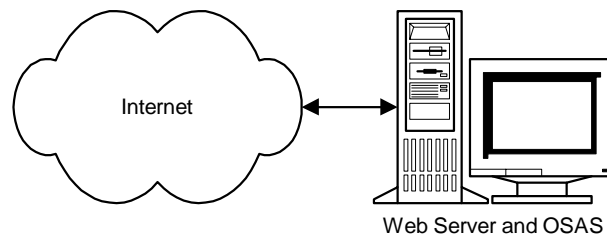
- creating your login page
- determining which inventory items are accessible
- granting access to your customers by assigning them passwords
- moving information between your OSAS system and the web server

Local and Remote Processing

Remote Processing



Local Processing



You can control how the data on your system is accessed when you set up OSAS Web B2B. You have two options regarding data access, *local processing* and *remote processing*.

Local processing allows direct access to your data files. This method provides the most timely information to your customers and is easy to set up and maintain.

Remote processing allows access only to copies of your data files. Remote process is more secure, allowing you to isolate the web server—and Internet users—from your other accounting data.

The method that you choose determines the tasks you need to perform to maintain the OSAS Web B2B system.

OSAS Web B2B Menu Structure

The OSAS Web B2B menu structure is similar to the structure of other OSAS applications: functions appear roughly in order of use.

Sales Order Processing

Use the functions on the **Sales Order Processing** menu to process and track the orders entered through the web server. You can produce a list of new orders and purge the order log. You can also copy the orders to your OSAS data files if you are using remote processing.

Remote Access

Use the functions on the **Remote Access** menu to copy data files and programs to the web server if you are using remote processing.

File Maintenance

Use the functions on the **File Maintenance** menu to establish groupings for Inventory item display, to add references to graphical pictures of items, and to set up and establish access to your item and customer information.

Setup OSAS Web B2B Components

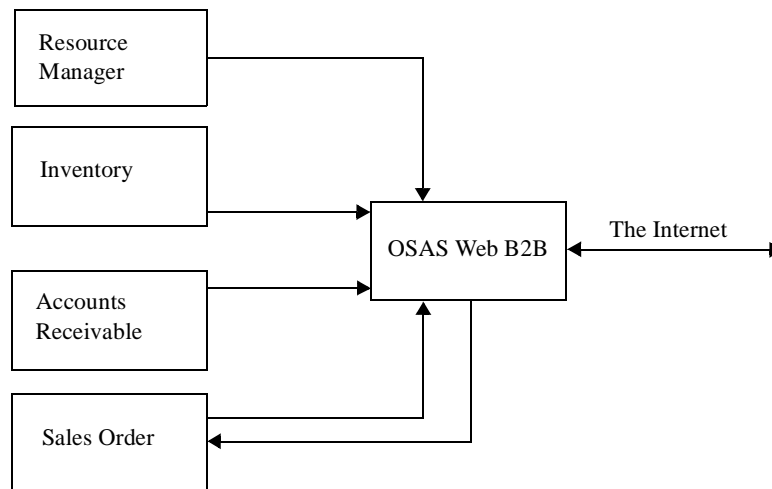
Use the functions on the **Setup OSAS Web Components** menu to create your Web Login Page based on the criteria you set up through the File Maintenance functions, and to install the OSAS Web components on your web server.

Master File Lists

Use the functions on the **Master File Lists** menu to produce listings of the information you entered using the File Maintenance functions. Use the lists to verify your entries or for future reference.

Application Interaction

OSAS Web B2B is an application that translates your OSAS data into fields accessible through the Internet. The information available for access comes from the Inventory, Accounts Receivable and Sales Order applications.



OSAS Web B2B's application interaction means that the information you enter in one application can be transferred to, and used in, other applications. This process reduces data entry time and the number of errors that might creep in along the way.

OSAS Web B2B uses data from Inventory, Accounts Receivable, and Sales Order, displaying this data to users on the Internet requesting the information from your web server. In return, sales orders entered through OSAS Web B2B update your Sales Order files.

You can control which files and data are available to Internet users.

Starting OSAS

OSAS runs on an operating system supported by 150 MB of permanent storage and 4 MB of RAM. You may need additional space or memory, depending on the size of your data files and the operating system you use. Consult your reseller for more information.

In Windows To start OSAS on a computer running Windows, double-click the OSAS shortcut on the desktop or access the program from the **Start** menu.

In Other Operating Systems To start OSAS on an operating system other than Windows, enter osas at the operating system prompt. If your operating system has graphical capabilities, you can also use the OSAS shortcut to start OSAS.

Using Parameters You can use the -u, -c, -a, and -t parameters in OSAS shortcut properties or after the **osas** command so that the system automatically uses the appropriate user ID, company ID, and access code to save time logging in.

In Windows, open the OSAS shortcut's properties and enter these parameters after the path in the **Target** field (as in the example below; be sure to use the correct directories for your system).

```
C:\basis\bin\bbj.exe osasstrt.txt -q -tT00 -cD:\osas70\progrm\config.bbx - -uSam  
-aapple -cH
```

Note: In Windows, the **-u**, **-c**, and **-a** parameters must follow the separation dash.

In other operating systems, enter the parameters after the osas command, as in this example:

```
osas -t T2 -c B -a apple
```

Note: You can enter these parameters in any order, but you must leave a space between the parameter mark (**-t**, **-c**, or **-a**) and the parameter itself.

Refer to the Resource Manager guide for more information on these parameters.

Logging In

After you start OSAS, the login screen appears.



To log in to OSAS, enter your **User ID**, the **Company ID** you want to work with, and your **Access Code**. If you want to save your access code so that you do not need to enter it again, select the **Save Password?** check box (or enter **Y** in text mode) to save your information. Finally, click **OK** or press **Enter** to log in.

This screen appears only after you have set up users and access codes for the OSAS system.


Access Codes

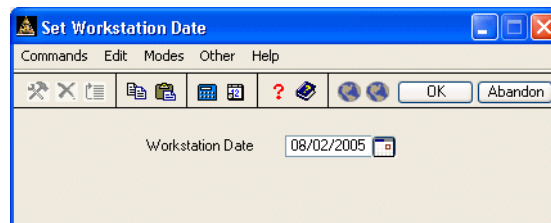
Access codes limit use of the system and protect sensitive information. Each code allows access to specific applications, menus, and functions. If you cannot select a menu or function, your access code is not authorized for it. Use the **Access Codes** function in Resource Manager to set up access codes.



To change access codes, select **Access code** from the **File** menu, click the **Access Code** button on the toolbar, or press **F4** on the main menu. When the Access Code box appears, enter the access code to change to and press **Enter**.

Workstation Date

 To change the workstation date, select **Workstation date** from the **File** menu, click the **Change Date** button on the toolbar, or press **F6**.



When the Workstation Date box appears, use the button or your keyboard to enter the date and press **Enter**.

Navigating OSAS

OSAS menus and functions are available in two modes: graphical and text. The graphical mode allows both keyboard and mouse commands and uses data entry fields and buttons similar to those found in any graphical software program. The text mode presents information in a simpler text format and uses keyboard commands to access functions and move around the screen. If you use an operating system that does not have graphical capabilities, the text mode is the only mode available.

You can use either text or graphical function screens independently of the main menu. For example, you can use text function screens while using the graphical main menu, and vice versa. Select **GUI Functions** from the **Modes** menu or press **Shift+F6** to toggle between the text and graphical modes for function screens.

When available, press **Shift+F5** to switch between graphical and text menu modes, or press **Shift+F6** to switch between modes on function screens. You can also use the Resource Manager **Defaults** function to select the default mode to use for the main menu and function screens.

In text mode, use the **Page Up**, **Page Down**, arrow, and **Enter** keys to move between menus, select and enter functions, and move around function screens. When a list of commands appears at the bottom of a function screen, press the highlighted letter to use a command. These methods also work in graphical mode, or you can use the mouse to click on fields and command buttons.

Graphical Mode

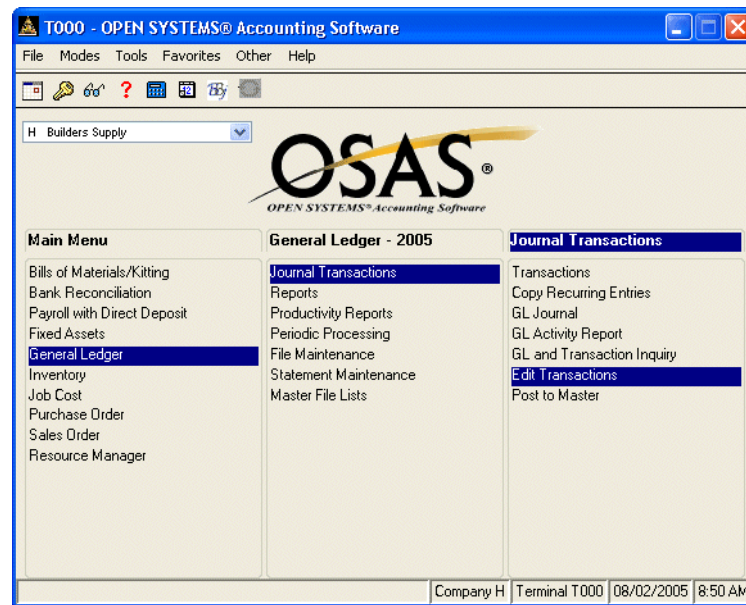
If you're familiar with other graphical software programs, you'll find it easy to navigate around the OSAS graphical mode, which uses buttons, toolbars, text entry boxes, and menus to help you move through your tasks.

Main Menu

If you use BBj in graphical mode, the main menu is available in two flavors: graphical and MDI. To switch between the two styles, press **Shift+F5**. If you use Visual PRO/5, the graphical main menu is the only graphical menu available.

Graphical Main Menu

The graphical main menu is shown below.

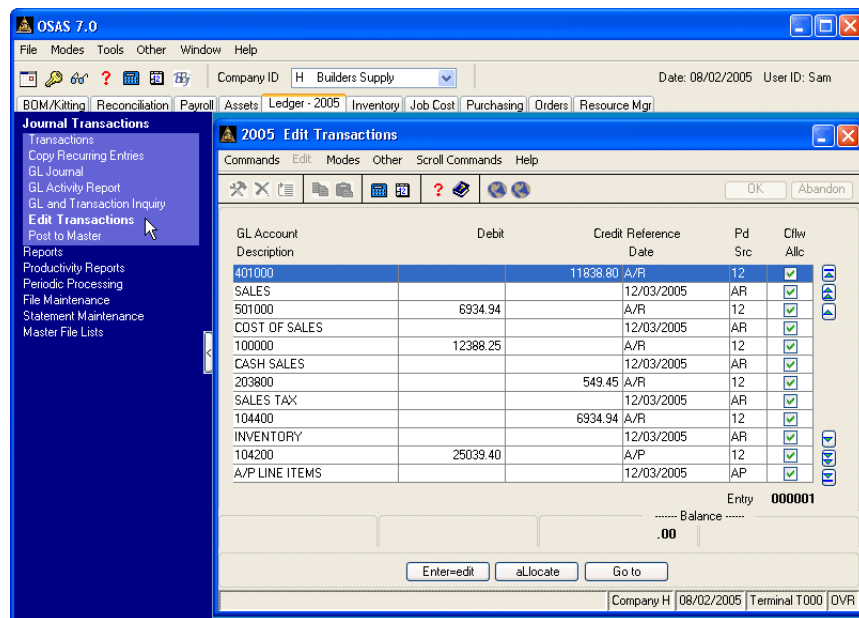


You can move around the graphical menu in these ways:

- Click an application to view that application's menu. Click a menu item to view its functions. Double-click a function name to enter that function.
- To exit from the graphical menu, click a different application or menu name or press **Tab** to return to the main menu.
- To exit from OSAS, click the **Close** box in the upper-right corner of the screen, press **F7**, or select **Exit** from the **File** menu.

MDI Main Menu

The MDI menu centralizes all OSAS functionality in one location: applications appear as tabs at the top of the screen, their menus and functions appear in a navigation pane on the left side of the screen, and function screens appear in the large pane on the right. Using this menu, you can open more than one function screen at a time and move or minimize screens as needed. However, you cannot open two functions that lock the same data file at the same time.



You can move around the MDI menu these ways:

- To view an application’s menus, click that application’s tab.
- To view the functions a menu contains, click the menu name. The menu expands to list the functions it contains. Click the function name to enter the function. The function screen appears in the right pane.
- To exit from a menu, click a different menu name or application tab. To exit from OSAS, click the **Close** box in the upper-right corner of the screen, press **F7**, or select **Exit** from the **File** menu.

Function Screens

Graphical screens contain the same functionality as text screens, presented in a graphical format that includes easy access to commands via the mouse.

Class	Description	Sales Tax	Purch Tax	Tax Collected	Tax Paid
00	Consumer Goods	6.500	6.500	1307.00	.00
01	Resale Sales	0.000	0.000	.00	.00
02	Exempt Sales	0.000	0.000	.00	.00
03	Ind/Agr Prod.	0.000	0.000	.00	.00
04	Interstate Comm	0.000	0.000	.00	.00
05	Motor Vehicles	0.000	0.000	.00	.00
06	Food Products	0.000	0.000	.00	.00
07	Clothing	0.000	0.000	.00	.00
08	Gasoline	0.000	0.000	.00	.00
09	Services	0.000	0.000	.00	.00
Total				1307.00	.00
Calculated				1307.01	.00
Over/Short				-.01	.00

You can move around the screen in these ways:

- Use the mouse or press **Tab** to move from field to field. Use the scroll buttons to move from line to line in scrolling regions.
- If a screen appears prompting for the kind of information to enter or maintain (such as on File Maintenance or Transactions screens), select the appropriate option and click **OK** to continue.
- Press **Page Down** if prompted to move to the next section.
- Click **Header** when it appears to return to the screen's header section.
- Press **F7** to exit the screen and return to the main menu.

Menus

Both the graphical main menu and graphical function screens contain drop-down menus that give you access to additional commands without using the function keys. While you can use the function keys to access commands in graphical mode, you may find it easier to access command through these menus.

To access a menu's commands, click a menu title. The commands for that menu appear, followed by any associated hot key combinations in brackets < >. To use a command, click the command name or press the hot key combination.

Refer to the Resource Manager guide for more information on the menus available in OSAS and their commands.

Shortcut Menu

OSAS gives you quick access to commands relating to the screen you're using via a shortcut menu. The commands that are available depend on the function and the field you are currently using. To use these commands, click the right mouse button and select the command from the menu that appears.

On the main menu, the shortcut menu gives you access to commands that help you manage your **Favorites** menu, switch between sample and live data, perform certain setup tasks, and view function information. On function screens, this menu helps you access help documentation, move around the function screen, work with EIS dashboards, and so on.

Other Commands Menu

The **Other Commands** (or **F4**) menu is available on both graphical and text menu and function screens and gives you access to additional utilities and commands not directly related to the function you're currently using. Among other things, these commands open calculators or allow you to view or enter additional information. In text mode, press **F4** twice on the menu or once on function screens to access this menu.

Consult Appendix A in the Resource Manager guide for more information on the commands available on the **Other Commands** menu.

Information Menu

The **Information** (or **Shift+F2**) menu is available in some graphical or text function screens in certain applications and gives you access to additional information about a customer, vendor, item, job, bill of material, or employee. The commands available on the **Information** menu are determined by the applications you have installed, and can include:

- General Information
- Comments
- History
- Documents
- Address Lookup

Not all of the commands above appear on every **Information** menu; instead, commands are available only as they are relevant to the task you are performing. For example, if you are entering a transaction in Accounts Receivable, you can access comments or documents about items or customers but not about employees or vendors.

Consult Appendix A in the Resource Manager guide for more information on how to use the functions on the **Information** menu.

Favorites Menu

The **Favorites** menu gives you quick access to the OSAS functions you use most by allowing you to add selections for entire menus or particular functions to a custom menu. After you've set up the menu, select **Change to Favorites** from the graphical **Favorites** menu or press **F2** to access the functions.

The **Favorites** menu saves you time by eliminating the need to switch between applications. You can add functions from several different applications to the **Favorites** menu and access them all there rather than switching between applications on the main menu to access the functions you need.

To add a function to the **Favorites** menu, select the function you want to add and press **F10**. Press **F2** to switch to the **Favorites** menu to confirm that your selection was added.

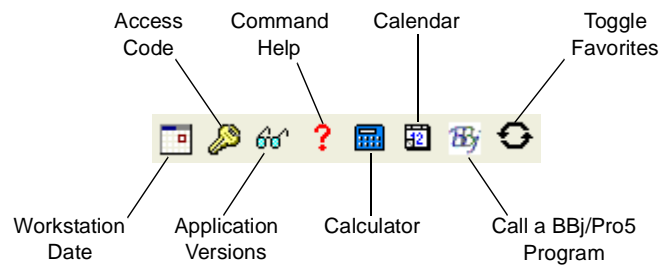
To remove a function from the menu, select the function on the **Favorites** menu that you want to remove and press **F10** again.

Toolbars

As with menus, graphical screens also contain toolbars that give you fast access to the most frequently used OSAS commands. The toolbar for the main menu differs slightly from that of function screens.

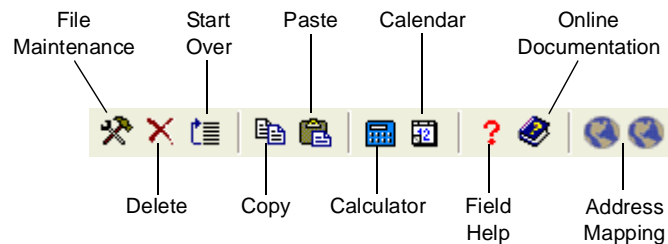
Main Menu Toolbar

The toolbar for the main menu is shown below. Click a button to access that command.




Function Screen Toolbar


The toolbar for function screens is shown below. Click a button to access that command.




Date Fields

 If you use BBj in graphical mode, click the **Calendar** button when it appears next to date fields to open a calendar so that you can select the date you want to enter into that field.


Browse

-  If you use BBJ in graphical mode, you can use the **Browse** button when it appears next to fields to navigate to directories and files and automatically enter file paths into that field. Click the **Browse** button to open the Select Directory/File screen, then navigate to the directory or file and click **Open** to automatically enter the file path in the field.


Inquiry

-  The Inquiry command helps you look up and select valid entries for fields that are connected to master file records. For example, when you use the Inquiry command in a **Batch ID** field, OSAS lists all batches you have set up so that you can select the one you want to enter in that field. When the **Inquiry** button appears next to a field, you can either click the button or press **F2** to open the Inquiry screen and search for valid entries.

Maintenance

-  The Maintenance command allows you to enter or edit master file records on the fly from within functions. For example, you can use the Maintenance command to add a new customer or item from within the **Transactions** function. The Maintenance command is available when the **Maintenance** button appears on the toolbar. Click the button or press **F6** to open the File Maintenance function associated with that field and enter or edit a new master file record.

Address Mapping

-  When you are working with a screen that contains an address, you can use the **Address Mapping** command to view a map of that address. This command combines address information with the URL and search variables in the Resource Manager **Web Setup** function and the **Map Lookup ID** in the **Company Setup** function to direct your web browser to a mapping website and generate the map.

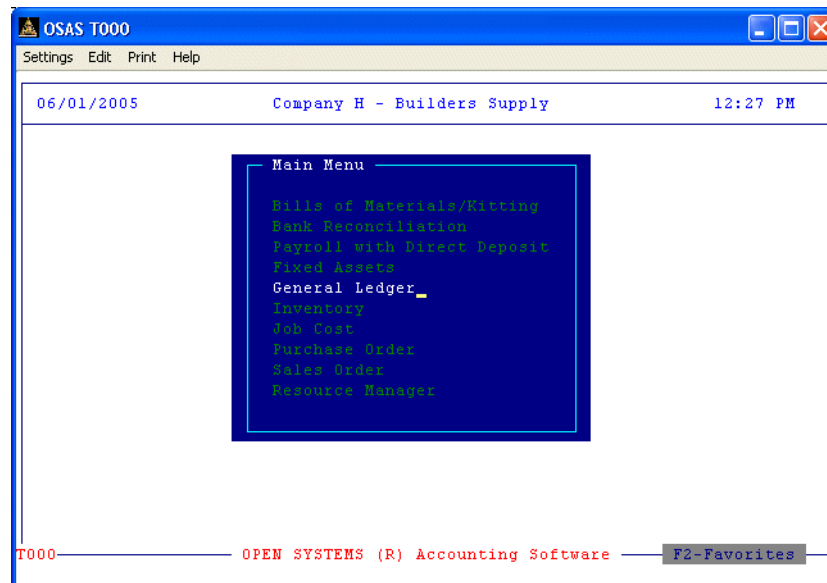
Note: Before you can view maps, you must set up mapping website information in the Resource Manager **Web Setup** function, select the **Map Lookup ID** to use in the Resource Manager **Company Information** function, and enter the path to your workstation's web browser in the Resource Manager **Defaults** function.

Text Mode

The OSAS text mode is available on all operating systems. If you use OSAS on an operating system that does not have graphical capabilities, the text mode is the only mode available. In text mode, all screens are presented in an easy-to-use textual interface that you navigate through using keyboard commands.

Main Menu

The text main menu is shown below.



When you select an application, the application's menu is superimposed over the main menu. Selecting an entry on an application menu opens a function screen or a submenu.

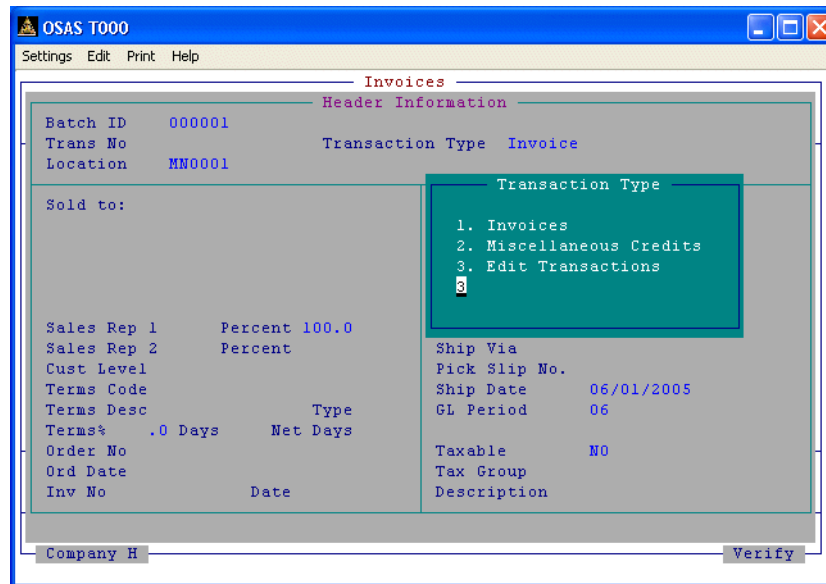
You can move around the text main menu in these ways:

- Use the arrow keys to move the cursor up and down to highlight the application you want. Then press **Enter** to select it.

-
- Press the first letter of the application you want to move the cursor to the first application beginning with that letter. Continue to press the letter key or the down arrow until the application you want is highlighted, then press **Enter** to select it.
 - Use the mouse to click an application to view that application's menu.
 - To move to the first application on the menu, press **Home**. To move to the last application on the menu, press **End**.
 - On an application menu, press **Page Up** to move to the menu immediately behind it. If you are several levels away from the main menu, you can return to the main menu by pressing **Page Up** repeatedly or by pressing **Tab** once.
 - To exit from OSAS, press **F7**.

Function Screens

Like the text menu, OSAS text function screens can be used on all operating systems and in combination with graphical menus.



You can move around the screen these ways:

- Press **Enter** or the down arrow to move from field to field.
- To use a command that is listed in the command bar, press the highlighted letter.
- Use hot key commands to access information screens or to toggle commands on and off. Refer to Appendix B in the Resource Manager guide for more information on these commands and their corresponding hot keys.
- If a screen contains more than one section, press **Page Down** when prompted to move to the next section.

-
- If a menu appears prompting you for the kind of information to enter or maintain (such as in the example and on Transaction and File Maintenance screens), select the appropriate option and press **Enter**.
 - To exit the screen and return to the menu, press **F7**.

Menus

Like the graphical mode, the text mode also includes menus that give you access to commands that open additional utilities, show additional information about the task at hand, or set up a custom menu that contains frequently-used commands.

Refer to Appendix A in the Resource Manager guide for full details about the menus available in OSAS.

Other Commands

The **Other Commands** (or **F4**) menu gives you access to additional utilities and commands not directly related to the function you're currently using. In text mode, press **F4** twice on the menu or once on function screens to access this menu. See page 1-19 for more information on this menu.

Information Menu

The **Information** (or **Shift+F2**) menu gives you access to additional information about a customer, vendor, item, job, bill of material, or employee. In text mode, this menu is available when the Info flag appears at the bottom of a function screen.

The commands on the menu are available only as they are relevant to the task you are performing. For example, if you are entering a transaction in Accounts Receivable, you can access comments or documents about items or customers but not about employees or vendors. See page 1-20 for more information.

Favorites Menu

The **Favorites** menu allows you add the OSAS menus or functions you use most frequently to a custom menu. After you've set up the menu, select **Change to Favorites** from the graphical **Favorites** menu or press **F2** to access the functions.

To add a function to the **Favorites** menu, select the function you want to add from the main menu and press **F10**. To remove a function from the menu, select the function on the **Favorites** menu that you want to remove and press **F10** again. See page 1-20 for more information on this menu.

Commands and Flags

Both the text menu and text function screens let you use commands to drill down to more information, change companies or access codes, switch to sample data, and perform tasks related to the function you are using. These commands are analogous to the commands contained on drop-down menus in graphical mode.

You access commands by pressing the hot key combination for the command you want to use. If you're working with a keyboard that lacks function keys (labeled with an **F** followed by a number) or if you're working with an emulator in UNIX (which can cause function keys to become unavailable), press the appropriate alternate key combination to access the command.

Refer to Appendix B in the Resource Manager guide for a list of all OSAS commands and their associated hot keys.

Not all commands are available for every function or field; when a command is available, a flag appears at the bottom of the function screen. Common flags include **Quick**, **Info**, **Maint**, **Inquiry**, and **Verify**.

- The **Quick** flag reminds you that you are using the Quick Entry mode to skip fields that are not required. Press **Ctrl+F** to toggle quick entry on and off.
- When the **Info** flag appears, press **Shift+F2** to access the **Information** menu to access additional information about a customer, vendor, item, job, bill of material, or employee. See page 1-20 for more information on this menu.
- When the **Maint** flag appears, press **F6** to launch the appropriate File Maintenance function to edit a master file record or enter a new one “on the fly.” When you finish, press **F7** to return to the function you were using.
- When the **Inquiry** flag appears, press **F2** to use the **Inquiry** command to look up additional information and select valid entries for the field you are in.
- The **Verify** flag reminds you that you are using verification. When this flag appears, you must provide verification when you press **Page Down** or use the **Proceed (OK)** command. Press **Ctrl+V** to toggle verification on and off.

Maint

Inquiry

Command Bar

The command bar appears at the bottom of function screen and gives you access to commands that allow you to move around the screen, add or edit information, change settings for selected lines, or select output devices.

```
Enter = edit, Append, Header, Totals, View, Online, Next trans
```

The commands that are available depend upon the function you are using, and are analogous to the command buttons available on graphical screens. Press the highlighted key to use a command.

Messages

Messages appear at the bottom of the screen when a command is unavailable or when OSAS needs information to continue.

```
Verification ————— Press <PgDn> to proceed
```

Address Mapping

When you are working with a screen that contains an address, you can use the **Address Mapping** command menu to view a map of that address. This command combines address information with the URL and search variables in the Resource Manager **Web Setup** function and the **Map Lookup ID** in the **Company Setup** function to direct your web browser to a mapping website and generate the map.

The **Address Mapping** command is available when the **Map** flag appears at the bottom of the screen. To view a map of the first address on the screen, press **Shift+F4**. To view a map of the second address (if present), press **Shift+F5**. The second command is not available when there is only one address.

Note: Before you can view maps, you must set up mapping website information in the Resource Manager **Web Setup** function, select the **Map Lookup ID** to use in the Resource Manager **Company Information** function, and enter the path to your workstation's web browser in the Resource Manager **Defaults** function.

Reports

All OSAS applications contain a variety of reports to help you view and analyze your business data. Each report function includes a selection screen that allows you to select the range of information to include in the report, which appears in alphabetical order when the report is produced. After you select the information to include, use one of these options to output the report:

- Select **Printer** (or enter **P** in text mode) to send the report to a printer, then select the printer to use.
- Select **Print Preview** (or enter **R**) to view the report in a preview window, from which you can print the report later. This option is only available on Windows or graphical Linux workstations running BBJ.
- Select **File** (or enter **F**) to save the report to a file, then change the directory path and file name (followed by the .txt extension), if necessary. Directory paths and file names must be less than 35 characters in length.

Note: To preserve formatting, view the reports you save to a text file with a fixed-width or monospaced font (Courier or Lucida Console, for example).

- In text mode, enter **S** to view the report directly in an OSAS function screen, then select whether to view it in Standard or Compressed width.
- When available, select **Email** (or enter **M**) to e-mail the report, then enter the e-mail address to sent the message to, the subject for the message, and whether to include the report as an attachment to the message.

Generally, reports or forms that make up part of your audit trail cannot be e-mailed. You also must set up your e-mail system in Resource Manager before you can e-mail reports.

Note: To preserve formatting, view e-mailed reports (or attachments) with a fixed-width or monospaced font (Courier or Lucida Console, for example).

Consult the Resource Manager guide for more information about reports.

Installation and Setup

2

Installing the Web Server	2-3
Installing OSAS Web B2B	2-5

Installing the Web Server

Before you install and set up OSAS Web B2B, you must install and configure your web server. OSAS Web B2B works with most popular web servers. However, the web server you use must have the ability to use Standard CGI.

You can install the web server on the same system on which OSAS is installed, or you may choose to install it on a separate computer. If you choose to use a different computer for a web server, you must be able to establish communications between your OSAS system and the web server through a network or modem connection. The web server and the OSAS system *must* be able to communicate because OSAS Web B2B provides several functions that move data files and programs from OSAS to the web server and back again.

When you set up your web server, you need a declaration in your CGI setup to handle BBx programs and graphical images. You must specify these virtual directories for

- CGI scripts (**/osasweb**) with the path to the OSAS Web B2B programs
- graphical images (**/image**) with the path to the graphical images
- style sheets (**/style**) with the path to the OSAS Web B2B programs and style sheets

If you have questions about setting up virtual directories, consult your web server user's manual.

The web server must have access to a BBx interpreter in order for OSAS Web B2B to function. If your web server can access your OSAS Resource Manager directory at all times, you can use the interpreter installed there for OSAS Web B2B. If not, you must install a BBx interpreter on the web server itself before you install OSAS Web B2B.

Once you have your web server installed and operational, you can proceed with OSAS Web B2B installation.

Security Procedures and Devices for OSAS Web B2B

Data security is a priority for many companies. OSAS Web B2B is designed to secure your files, customer lists, and account information through use of integral password checks which you setup during your OSAS Web B2B installation. To enhance OSAS Web B2B's built in security codes, you may install a *firewall* for your web server.

A firewall is a generic name for a program or system that enforces access control between two network systems. By requiring any user to enter a password, a firewall acts as a buffer from unwanted access into your system. While they do not protect against viruses, firewalls are recommended to protect sensitive data.

If you have questions and concerns, your value added reseller can supply you with answers and suggestions.

Note

If you are using a Windows Web Server that accesses data through a UNIX Data Server, see page 4-7.

Installing OSAS Web B2B

When you set up OSAS Web B2B, you can elect to allow access to your OSAS data files directly from the Internet, or to allow access to copies of your files stored on the web server (even if the web server is installed on the same machine). The process of using copies of your files is known as *remote processing*.

If you elect to use remote processing, you must complete a few additional steps after you complete the installation and setup checklist below.

OSAS Web B2B Requirements and Applications

You must have the Accounts Receivable application installed and set up on your system to use OSAS Web B2B. OSAS Web B2B uses your Customer file as a basis for the access codes that allow your customers to access the OSAS Web B2B functions on the Internet.

You have control of which OSAS Web B2B functions can be run, and by which customers. However, some of the functions require other OSAS applications in order to work.

With Accounts Receivable installed, your customers can use the **Account Information, History Inquiry, Invoice Inquiry, and Aged Trial Balance** functions.

If Inventory is installed on your system, your customers can access the **Item Inquiry** function.

If Sales Order is installed, your customers can use the **Order Inquiry** function.

If you have both Inventory and Sales Order installed, your customers can use the **Sales Order Entry** function.

OSAS Web B2B Setup Checklist

Follow these steps to install and set up OSAS Web B2B on your computer:

Note

Many of the functions you use during the setup process ask you for path (or directory) names where certain web server files, programs and data files reside. Read the instructions for these functions carefully. Some path names describe the directory from the web server to the OSAS system, and others describe the directory from the OSAS system to the web server.

1. Use **Resource Manager** to install OSAS Web B2B on your computer. See the **Resource Manager** guide for installation instructions.
2. Use the **Company Information** function in **Resource Manager** (see the **Resource Manager** guide) to set up each company for which you want to provide web access.
3. Use the **Customer Internet Access Codes** function (page 3-13) to control which of your customers can access the OSAS Web B2B functions, the functions to which each customer has access, and the **Inventory Location** that is used to calculate prices for this customer.
4. Use the **Item Group Codes** function (page 3-3) to set up item groups for item inquiry and sales order entry web functions.
5. Use the **Internet Inventory Item Groups** function (page 3-9) to add items or item subgroups to the groups you set up.
6. Use the **Inventory Item Pictures** function (page 3-5) to associate graphic images with inventory items for display in the item inquiry and sales order entry web functions.

-
7. Use the **Customer Groups** function (page 3-17) to set up the item groups to which each customer has access during the item inquiry and sales order entry web functions.
 8. Use the **Tables** function (page 3-21) to set up the **EMAIL** table, which stores the E-mail address your customers use to contact you if they encounter problems using the web functions.
 9. Use the **Create Login Page** function (on page 4-3) to build your login page and to copy it to your web server directory.
 10. Use the **Install Web Server Components** function (on page 4-5) to set up the communication between the web server and OSAS, and to copy the OSAS Web B2B programs and drivers to the web server.

Note

You may need to edit the CONFIG.BBX file created on the web server after completing the **Install Web Server Components** function. Consult Appendix D for more information on modifying the CONFIG.BBX file.

Follow these additional steps if you are using remote processing with OSAS Web B2B:

11. Use the **Copy OSAS Programs to Web Server** function (page 6-7) to copy certain special OSAS programs to the web server.

These programs perform tasks such as accessing inventory quantity and pricing information, calculating balances, and so on. If your web server can access the Resource Manager programs path directly at all times, you can skip this step.

12. Use the **Copy Data Files to Web Server** function (page 6-5) to copy your data files to the web server. You must copy the optional data files when setting up OSAS Web B2B.

Note

Using copies of your data files may provide a higher level of security, but it requires you to update the files periodically on both the OSAS system and on the web server. You can update files using the functions on the OSAS Web B2B menus. See chapter 5 for more information about these periodic tasks.

File Maintenance

3

Item Group Codes	3-3
Inventory Item Pictures	3-5
Internet Inventory Item Groups	3-9
Customer Internet Access Codes	3-13
Customer Groups	3-17
Tables	3-21
Change Fields	3-23

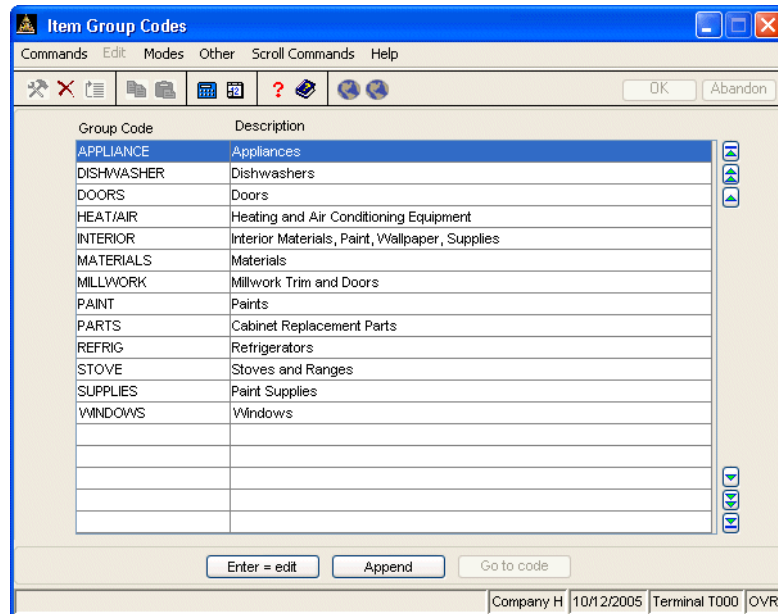
Item Group Codes

Use item groups to group inventory items together for display and security purposes on the Item Inquiry and Order Entry web pages. You can restrict access to the items by the groups you set up.

Use the **Item Group Codes** function to add item groups or change the description of existing item groups. The groups you create are used to restrict access to your inventory files. You can place the same inventory items in multiple groups, or exclude inventory items from all groups, as you like. You can then specify the groups to which each customer has access.

Item Group Codes Screen

The screen lists the group codes and their descriptions.



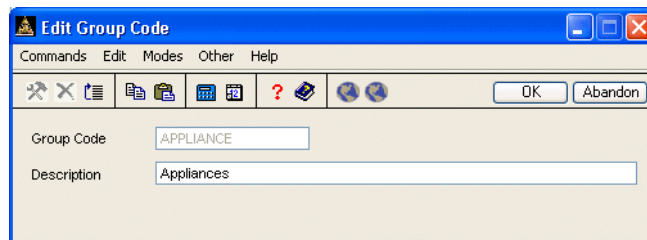
Use these commands to work with the group codes:

- Press **Enter** to edit the selected item group code description. The Edit Group Code screen appears.
- Press **A** to add a new item group to the list. The Append Group Code screen appears.
- Press **G** to go to a specific item group code, then enter the code. This command is available only when there is more than one screen of item group codes.

When you finish, use the **Exit (F7)** command to return to the **File Maintenance** menu.

Append/Edit Group Code Screen

The Append Group Code Screen appears when you add a new group code to the list. The Edit Group Code Screen appears when you edit an existing group code. Other than the title, these screens are identical.



Enter the group code (up to 10 characters). If you are editing an existing group code (as in the example above), you cannot change the group code. Instead, return to the Group Codes screen, use the **Delete (F3)** command to delete the incorrect code, then press **A** to enter a new code.

Enter the code's description, then use the **Proceed (OK)** command to save your changes and return to the Group Codes screen.

Inventory Item Pictures

Use the **Inventory Item Pictures** function on the **File Maintenance** menu to associate a graphic file with your inventory items. The graphic image associated with the item appears on the Item Inquiry and Order Entry web pages.

You can use a variety of graphic image formats, including **.JPG** and **.GIF** image files.

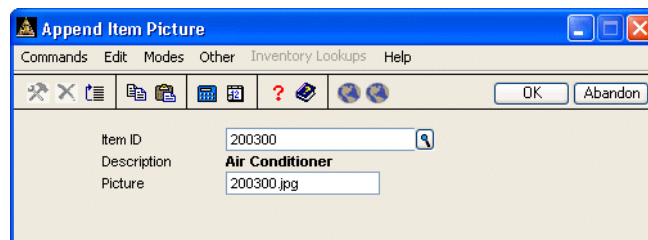
Not all graphics formats can be viewed with all browsers. Generally, **.GIF** and **.JPG** images are recommended because most browsers can display them, and because their small file size allows them to load faster.

If you do not associate a graphic file with an inventory item, OSAS Web B2B uses the convention of item ID + **.GIF** to search for images. This convention can save you time. Instead of taking the time to associate graphic image files with items using this function, name your images with the appropriate item ID and place them in the directory specified for the virtual **/image** directory on the web server. When a customer accesses a web page that uses item images, OSAS locates the appropriate graphic file according to the convention and displays it on the page.

In order to appear correctly on the web page, the image files must be located in the directory specified for the virtual **/image** directory on the web server.

Append/Edit Item Picture Screen

The Append Item Picture screen appears when you add a picture to a new item. The Edit Item Picture screen appears when you edit an existing picture. Other than the title, these screens are identical.



Field	Value
Item ID	200300
Description	Air Conditioner
Picture	200300.jpg

Inquiry

Enter the item ID. The item's description appears. If you are editing an existing picture, you cannot change the item ID. Instead, return to the Item Picture screen, use the **Delete (F3)** command to delete the item from the item picture file, and then press **A** to add a new picture.

Enter the picture's file name, then use the **Proceed (OK)** command to save your changes and return to the Inventory Item Pictures screen.

Internet Inventory Item Groups

Use the **Internet Inventory Item Groups** function to:

- Add inventory items to an item group.
- Add subgroups to an item group.
- Change the inventory locations included in an item group.
- Remove inventory items or locations from an item group.

Item groups organize inventory items together for display and security purposes on the Item Inquiry and Order Entry web pages. You can restrict access to the items by the groups you set up.

Use the **Item Group Codes** function (page 3-3) to add item groups to the system.

Internet Inventory Item Groups Screen

Sub	Item ID/Subgroup Code	Description	Loc. ID	Picture
<input checked="" type="checkbox"/>	400	Interior Materials	CA0001	
<input type="checkbox"/>	400	Interior Materials	MD0001	
<input type="checkbox"/>	400	Interior Materials	MN0001	
<input type="checkbox"/>	400	Interior Materials	TX0001	
<input type="checkbox"/>	800001	Wallpaper - Contemporary	CA0001	
<input type="checkbox"/>	800001	Wallpaper - Contemporary	MD0001	
<input type="checkbox"/>	800001	Wallpaper - Contemporary	MN0001	
<input type="checkbox"/>	800001	Wallpaper - Contemporary	TX0001	
<input type="checkbox"/>	800002	Wallpaper - Traditional	CA0001	
<input type="checkbox"/>	800002	Wallpaper - Traditional	MD0001	
<input type="checkbox"/>	800002	Wallpaper - Traditional	MN0001	
<input type="checkbox"/>	800002	Wallpaper - Traditional	TX0001	
<input checked="" type="checkbox"/>	PAINT	Paints		
<input checked="" type="checkbox"/>	SUPPLIES	Paint Supplies		

Inquiry**Maint**

After you select a group code, the screen lists the subgroups and items set up for that code. Only the group codes you set up in the **Item Group Codes** function (page 3-3) appear in the **Group Code** field.

If you have set up a picture for an item using the **Inventory Item Pictures** function (page 3-5), the graphics file name appears in the **Picture** field.

The **Sub** field indicates whether the line item is a subgroup: if the check box is selected (or if **YES** appears in text mode), the line item is a subgroup; if it is clear (or if **NO** appears in text mode), the line item is an inventory item.

To set up a subgroup, first set up the subgroup code and description using the **Item Group Codes** function (page 3-3). Next, enter that code in the **Group Code** field and add inventory items to it. Finally, enter that group code as a line item for a different group code. When the **Is this item a subgroup?** message appears, select the check box (or enter **Y** in text mode) to indicate that you are adding a subgroup, then enter the subgroup code.

To exclude an item (or item location) from the list, select the line to exclude and use the **Delete (F3)** command to remove it from the list.

Commands

Use these commands to work with the items and subgroups for a group code:

- Press **F** to view items for the first item group on file.
- Press **N** to view items for the next item group on file.
- Press **P** to view items for the previous item group on file.
- Press **S** to view items for the last item group on file.
- Press **A** to add an item or subgroup to the group. The **Is this item a subgroup?** message appears.

Inquiry

To add a subgroup to this group, select the check box (or enter **Y** in text mode) and then enter the subgroup code.

Inquiry

To add an item to this group, clear the check box (or enter **N** in text mode) and then enter the item ID. All locations for the item are added to the list.

- Press **L** to include all inventory items in this item group.
- Press **O** to remove all inventory items from this group.
- Press **H** to select a different group code.
- Press **R** to add a range of item IDs to the group. The Add Range of Items screen appears.
- Press **G** to move to a specific item, then enter the item ID or use the **Inquiry (F2)** command to select the item from the list that appears. This command is available only when there is more than one screen of items.

To save your entries, press **Enter** at the **Item ID** or **Subgroup Code** field. To exit to the **File Maintenance** menu, use the **Exit (F7)** command.

Add Range of Items Screen

The Add Range of Items screen appears when you press **R** on the Internet Inventory Item Groups screen to add a range of items to an item group.

Inquiry

Select the range of item IDs, locations, and product lines to add to the inventory item group. To add all items for a location to an item group, enter only the location IDs whose items you want to add. To add all items from an inventory product line to a group, enter only the product lines.

Customer Internet Access Codes

You grant Internet access to your data for your customers through Internet access codes. You can limit access for certain customers to particular web pages and item groups.

You can set up multiple access codes for specified customers. This function allows some of your customers' employees access to limited data, while others can access more extensive information.

You can also set up *master access codes* that are not associated with a particular customer. For example, these access codes can be given to your sales representatives, allowing them remote access to multiple customers' data. Master passwords cannot match any customer-specific passwords you set up.

Use the **Customer Internet Access Codes** function to:

- Add access codes for a customer.
- Change the information that a customer can access with an access code.
- Set up additional access codes for a customer.
- Set up master passwords for your staff.
- Remove access for customers.
- Set up the inventory location to use when calculating prices for a customer.

Note

You must set up access codes for your customers before they can access the information on your web site.

Customer Internet Access Codes Screen

Options like these give customers access to the type of information listed for that web page.

The **Link to Order Entry** options give customers access to the Order Entry web pages from within the listed web page.

Select this option to show customers on hand inventory item quantities on the Order Entry web page.

Select this option to show customers the message "Available" on the Order Entry web page when on hand quantities exist in Inventory. You must elect to show on hand quantities before you can use this option.

If no on hand quantities exist in inventory, the message "Out of Stock" appears instead.

Inquiry

1. Enter the customer ID to which you want to give access. To set up a master password, leave this field blank.

Inquiry

2. Enter the access code. A customer may have multiple access codes.

Inquiry

3. If you enter a new access code, the **Copy From Company ID** field appears. You can copy access parameters from a customer in this company or in another company. Enter the company ID from which you want to copy customer access codes, or press **Enter** to proceed without copying.

Inquiry

4. If you entered a company ID, the **Copy from Customer ID** field appears. Enter the customer ID from which you want to copy an access code. The customer ID you specify must use the same access code you specified in the **Access Code** field.

Inquiry

5. Enter the inventory location ID you want to use to calculate prices for this customer.
6. If the customer has access to the listed web page or its related information, the **Access** check box is selected (or **YES** appears in text mode). If the customer does not have access, the box is clear (or **NO** appears in text mode).

Press **Enter** to toggle access on and off for the selected option, or use the commands to toggle access for all options.

7. Use these commands to work with the HTML page options in the list:
 - Press **Enter** to toggle access for the selected option on and off.
 - Press **L** to give the customer access to all the page information listed.
 - Press **N** to deny the customer access to all the page information listed.
 - Press **W** to save your changes.
 - Press **C** to return to the **Customer ID** field to select another customer.
 - Press **G** to open the Customer Groups screen (page 3-18) where you can specify which item groups this customer can access.
 - Press **D** to remove this access code for this customer.
 - Press **A** to add an HTML page option to the list.
 - Press **E** to edit an HTML page option. You cannot edit system pages.
8. To save your entries, press **W**. Next, enter another customer ID and access code or use the **Exit (F7)** command to return to the **File Maintenance** menu.

Customer Groups

Use the **Customer Groups** function to:

- Add customer access to an item group.
- Change the customers that have access to an item group.
- Remove customer access from an item group.

You can work with customers and item groups in two ways. You can either:

- Specify a group and enter the customers that have access to it.
- Specify a customer and enter the groups the customer can access.

Customer Groups (Item Group) Screen

Customer ID	Customer Name
ACE001	ACE BUILDERS
CASHCA	CASH SALES-OAKLAND, CA
CASHMD	CASH SALES-BALTIMORE, MD
CASHMN	CASH SALES-MINNEAPOLIS
CASHPS	CASH SALES-DALLAS, TX
CASHTX	CASH SALES-DALLAS, TX
DAL001	DALLAS-FT WORTH DOME HOMES
GRE001	GREATER NEW YORK DOMES, INC.
KAN001	KANSAS CITY GEODESIC HOMES
LOS001	LOS ANGELES CONSTRUCTION CO.
SUN001	SUNSHINE HOMES, INC.
TEN001	TENNESSEE SHELTERS, INC.
VIS001	VISA

Inquiry
Maint

To list the customers that have access to a specific group code, enter the code in the **Group Code** field. The customers with access to that group appear.

Customer Groups (Customer) Screen

Group Code	Group Description
APPLIANCE	Appliances
DOORS	Doors
HEAT/AIR	Heating and Air Conditioning Equipment
INTERIOR	Interior Materials, Paint, Wallpaper, Supplies
MATERIALS	Materials
MILLWORK	Millwork Trim and Doors
PARTS	Cabinet Replacement Parts
WINDOWS	Windows

Inquiry

Maint

To list the group codes a specific customer can access, leave the **Group Code** field blank and enter the customer ID you want to view in the **Customer ID** field. The item groups this customer can access appear.

Use these commands to work with the information on the screen:

- Press **F** to view the first item group or customer on file.
 - Press **N** to view the next item group or customer on file.
 - Press **P** to view the previous item group or customer on file.
 - Press **S** to view the last item group or customer on file.
- Inquiry
- Press **A** to add a customer or item group to the list, then enter the ID or code to add.

-
- Press **L** to add access for all customers or item groups to the list.
 - Press **O** to clear the list for the selected item group or customer.
 - Press **H** to return to the header section to select a different customer or item group.
 - Press **G** to go to a specific customer or item group in the list, then enter the ID or code. This command is available only when there is more than one screen of information.

Tables

Tables store information and options used by the system. Use the **Tables** function to set up and maintain the OSAS Web B2B **EMAIL** table. The **EMAIL** table stores the address of the contact person for web problems. This address appears on the web pages as the customer's contact if any problems occur while viewing information.

OSAS Web B2B also uses these tables:

- The **HOMEPATH** table stores your entries when you use the **Create OSAS Web Login Page** function.
- The **WEBPATH** table stores your entries when you use the **Install Web Server Components** function.
- The **COPYPATH** table stores your entries when you use the **Copy Programs to Web Server** and **Copy Data Files to Web Server** functions.
- The **IMGPATH** table stores the path to the directory where you store image files for use by the web server.

Note

The **HOMEPATH**, **WEBPATH**, **COPYPATH**, and **IMGPATH** tables are automatically created and maintained by the functions described above. You should not change these tables using the **Tables** function. Instead, change the information in these tables using the functions listed. Use the **Tables** function to maintain only the **EMAIL** table.

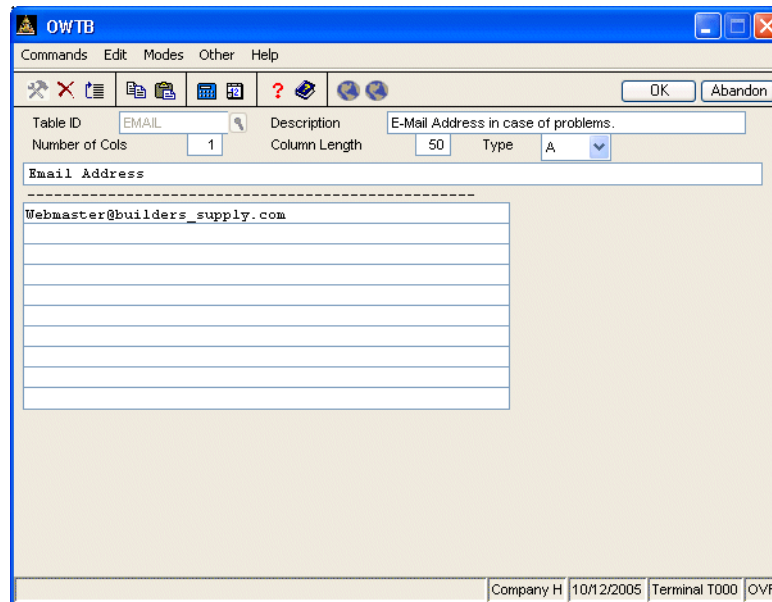
Tables Screen

Follow these instructions to maintain the **EMAIL** table using the **Tables** function:

1. Select **Tables** from the **File Maintenance** menu. A blank Tables screen appears.

Inquiry

2. Enter **EMAIL** in the **Table ID** field. The **EMAIL** table's description, number of columns, column length, type (alphanumeric or numeric), headings, and data appear.



The screenshot shows a window titled "OWTB" with a menu bar (Commands, Edit, Modes, Other, Help) and a toolbar. The main area displays the configuration for the "EMAIL" table. The "Table ID" field contains "EMAIL", and the "Description" field contains "E-Mail Address in case of problems.". Below these are fields for "Number of Cols" (1), "Column Length" (50), and "Type" (A). A table with the heading "Email Address" is shown, containing one row with the value "Webmaster@builders_supply.com". The status bar at the bottom indicates "Company H", "10/12/2005", "Terminal T000", and "OVR".

3. Enter the e-mail address of the person in your company who should be contacted in the event of web problems.
4. Use the **Proceed (OK)** command to save your changes. The cursor returns to the **Table ID** field. Use the **Exit (F7)** command to return to the **File Maintenance** menu.

Header

Inquiry

1. Enter the **Field ID** you want to change. You can change only General Ledger fields from the **General Ledger** menu. To change IDs and codes from other applications, run the **Change Fields** function in the respective application.
2. Select the **Print Log?** check box to print a list of the files that are changed.
3. After you enter the **Field ID** and indicate your preference for printing the log, use the **Proceed (OK)** command to begin entering field values to change.

Values

4. To edit or add original/new values in this section, select a line and press **Enter** to edit the current line. The Edit Original/New Values dialog box appears. Press **A** to append another value to the list. The Add Original/New Values dialog box appears.
5. Enter the current field value you want to change in the **Original Value** box.
6. Enter the new value that you want to use for this field in the **New Value** box.
7. Select a command.
 - Press **S** to switch to the **File Description** section to specify which files change during processing.
 - Press **Enter** to edit the current line.
 - Press **A** to append another value to the list.
 - Press **B** to begin the change field process.
 - Press **H** to return to the header section to change the selection you made for printing the log.

-
- Press **G** to go to a particular entry. This option is only available when there is more than one page of entries.
 - Press **F** to choose a new field ID (this abandons any field changes you entered, but have not yet saved).
8. Continue entering old values and new values until you have specified all of the values you want to change in the **Values** section.

Files

The files that contain the **Field ID** you selected appear in the **File Description** section. You should change IDs in all of the files as a general rule. Exclude files from the change process only when your reseller or support representative instructs you to so.

9. The **Time** field gives you an idea of the relative time it takes to change the field in a given file. Files where this code or ID are a part of the key to the file can be changed more quickly than files where each record in the file must be scanned for the code or ID. Each file is rated as **Short** or **Long** to denote the estimated time required to change the field.
10. The **Tag** field denotes whether the file is affected by the copy process. Tag the file to change fields in the file.
11. Select a command.
 - Press **S** to switch to the **Values** section of the screen.
 - Press **Enter** to toggle a file as included or excluded from the copy process.
 - Press **A** to tag all of the files.
 - Press **N** to untag all of the files.
 - Press **B** to begin the change field process.

- Press **H** to return to the header section to change the selection you made for printing the log.
 - Press **G** to go to a particular entry. This option is only available when there is more than one page of entries.
 - Press **F** to choose a new field ID (this abandons any field changes you entered, but have not yet saved).
12. When you have tagged the files you want to change, press **B** to begin the change process. When the changes are complete, the log prints if you elected to produce it.
 13. Enter a new **Field ID** to change, or use the **Exit (F7)** command to return to the **File Maintenance** menu.

Change Fields Log

10/12/2005	Builders Supply				Page	1
3:02 PM	Change Field Log					
File Name	Records Read	Records Converted	Original Total Record	New Total Records		
OWGCH	4	2	13	13		
OWCGH	3	1	26	26		
OWICH	18	16	220	220		
Field ID	OW GROUP CODE					
Original Value			New Value			
APPLIANCE			APPS			
DISHWASHER			DSW			

Set Up Web Components

4

Create Login Page	4-3
Install Web Server Components	4-5

Create Login Page

Use the **Create Login Page** function to build the login page for the web server. You must run this function to:

- create your OSAS Web B2B login page for the first time.
- change the title used on the login page.

Note: Before you can use this function, make sure the company's **OSAS Web** field is set to **Active** in the Resource Manager **Company Information** function.

Create Login Page Screen

The screenshot shows a dialog box titled "Create Login Page". It features a menu bar with "Commands", "Edit", "Modes", "Other", and "Help". Below the menu bar is a toolbar with icons for back, forward, search, and help, along with "OK" and "Abandon" buttons. The main area contains two text input fields: "Login Page Title" with the value "Builders' Supply" and "Web Server Documents Path" with the value "C:\inetpub\wwwroot/". At the bottom of the main area are three labels: "Building Login Header", "Building Login Body", and "Building Login Trailer". The status bar at the bottom right shows "Company H | 10/12/2005 | Terminal T000 | OVR".

Enter a title for your login page, then enter the directory path name to the location on the web server from which web pages are launched.

To save your entries and create the login page, use the **Proceed (OK)** command. After the page is created, the **Set Up Web Components** screen appears.

Install Web Server Components

Use the **Install Web Server Components** function to copy the components required for OSAS Web B2B to the web server. This function copies the various OSAS Web B2B web pages and web drivers.

The information copied to the web server is divided into three categories:

- Web server programs, including the OSAS Web B2B programs, scripts, style sheets, and batch files.
- Data files, including the OSAS Web B2B, Accounts Receivable, Sales Order and Inventory files, copied by the **Copy Data Files to Web Server** function (page 6-7).
- Resource Manager programs, including the applications programs used in calculating prices, aging customers, and so on, copied by the **Copy Programs to Web Server** function (page 6-7).

In this function, you tell OSAS Web B2B where this information is stored. You need to describe how to reach the web server from the OSAS system, and how to access the information from the web server.

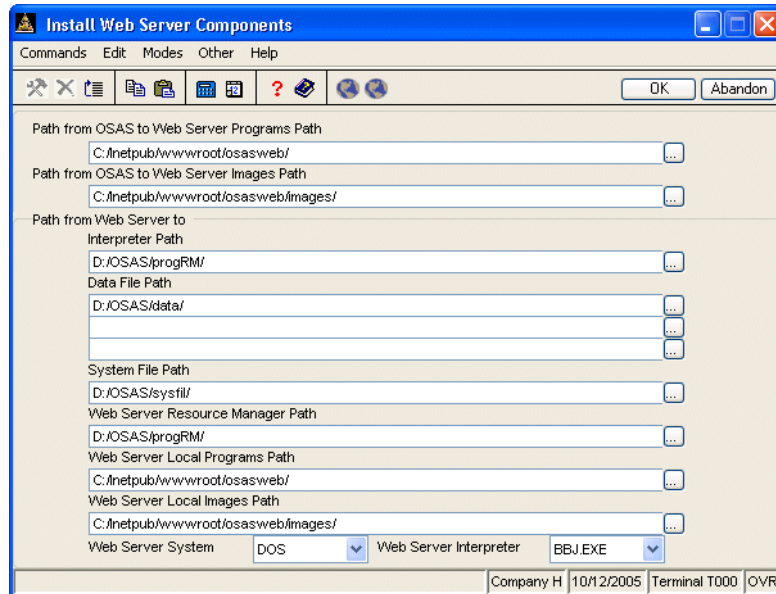
If you use local processing, the programs and data files reside on the same system where OSAS is installed. If you use remote processing, the programs and data files reside on the web server.

Note

If you are using a Windows Web Server that accesses data through a UNIX Data Server, see page 4-7.

For a list of the files copied, see appendix C.

Install Web Server Components



Field Definitions

Field Name	Description
Web Server Programs Path	Enter the full path from the OSAS system to the web server programs path.
Web Server Images Path	Enter the full path from the OSAS system to the web server images path.
BBx Interpreter Path	Enter the full path from the web server to the location of the BBx interpreter.
Data File Path	Enter the full path from the web server to the OSAS data files. If you use remote processing, enter the path where the data file copies are kept.

Field Name	Description
System File Path	Enter the full path from the web server to the OSAS sysfil path. If you use remote processing, enter the path where the sysfil copies are kept.
Web Server Resource Manager Path	Enter the full path from the web server to the location of the Resource Manager programs.
Web Server Local Programs Path	Enter the full path to the directory mapped to the virtual osasweb directory on the web server.
Web Server Local Images Path	Enter the full path to the directory mapped to the virtual images directory on the web server.
Web Server System	If the web server resides on a UNIX/Linux system, enter U . If the web server resides on Novell or Windows 95/98 or NT, enter D .
Web Server Interpreter	Select the type of interpreter the web server uses, BBJ.EXE or VPRO5.EXE .

To save your entries or begin the copy process, use the **Proceed (OK)** command. When the copy process finishes, the **Set Up Web Components** menu appears.

Windows Web Server/UNIX Data Server

If you are using a Windows Web Server that accesses data through a UNIX Data Server, you must set up a UNIX user ID on the UNIX Data Server.

On the UNIX computer accessing the OSAS Web B2B data files, set up **osasweb** as a UNIX user ID.

Note: If you do not want to use **osasweb** as your user ID, you must change the userid entries in the **OW*.tmp** files in the **progOW** directory. Edit the **OW*.tmp** files **before** you use the **Install Web Server Components** function.

Sales Order Processing

5

Sales Order Processing on the Web	5-3
Transaction Journal	5-5
Build Sales Orders from Remote Files	5-7
Purge Log File	5-11

Sales Order Processing on the Web

If you allow your customers access to the sales order processing functions on the web, use the functions on the **Sales Order Processing** menu.

When you set up OSAS Web B2B, you can elect to allow access to your OSAS data files directly from the Internet, or to allow access to copies of your files stored on the web server. The process of using copies of your files is known as *remote processing*. See chapter 2 for more information about remote processing.

When your customers enter sales orders through the Internet, OSAS Web B2B stores the orders as quotes in the Sales Order Header and Detail files. If you are using remote processing, OSAS Web B2B uses the files kept on the web server. If not, OSAS Web B2B uses the same files used in Sales Order Processing. In either case, OSAS Web B2B keeps a log file of the orders that have been entered, and any comments customers enter along with their orders.

Use the functions on the **Sales Order Processing** menu on a daily basis, as described below (skip steps 2 and 4 if you do not use remote processing):

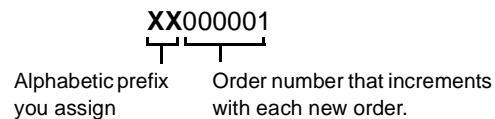
1. Print the Transaction Journal to list the orders that have been entered along with any comments from the customers.
2. If you use remote processing, run the **Build Sales Orders from Remote Files** function to transfer your customers' orders from the remote data files to your live OSAS data files.
3. Use the **Purge Log File** function to prepare the log file for the next day's entries.
4. If you use remote processing, run the **Copy Data Files to Web Server** function on the **Remote Access** menu.

Order Numbers

By default, OSAS Web B2B uses the numbers set up in Sales Order to number sales orders entered through the web. When a customer enters an order through the web, the system determines the next available order number and enters the order as a price quote with that order number.

If you use remote processing, the system scans for available numbers a second time when you build sales orders from remote files in order to prevent overwriting quotes entered internally through Sales Order. Any changed order numbers print on a log after web orders are copied to your live system.

You may prefer to set up order numbers to easily distinguish between orders entered internally and those entered through the web. Using the **Company Information** function on the Resource Manager **Company Setup** menu, you can assign a two-character prefix to order numbers for web orders. OSAS then uses this convention to assign order numbers:



If you assign only a one-character prefix, the remaining numeric portion contains seven numbers instead of six.

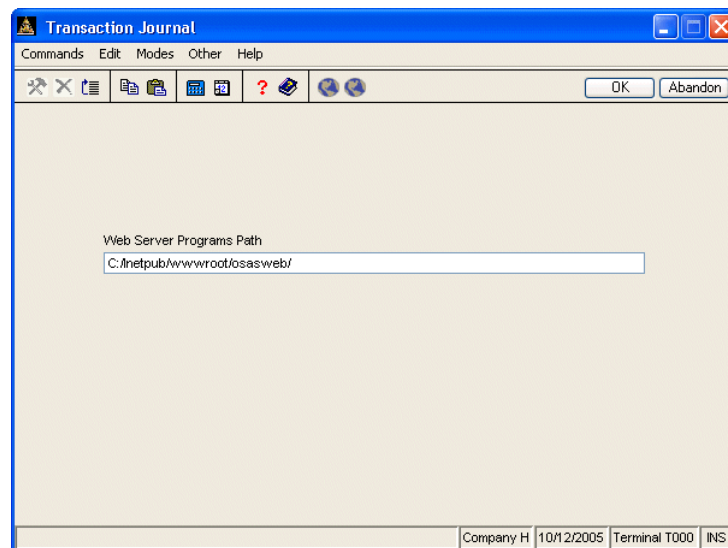
When you use prefixes, orders entered through the web appear in Sales Order as price quotes with order numbers in the format described above (even if you use remote processing). Use these prefixed order numbers to quickly distinguish between quotes entered via the web and those entered internally.

Transaction Journal

The **Transaction Journal** function produces a list of the orders entered on the web site. This list serves as an audit trail of the transactions, as well as an edit register to check for necessary changes or incomplete entries.

If you use remote data files, you can print the Transaction Journal before moving the orders to your live Sales Order files as an audit trail. When you use the **Build Sales Orders from Remote Files** function (page 5-7), this report reprints automatically to show any order number changes that have been made in the copy process.

Transaction Journal Screen



1. The location of the OSAS Web B2B directory on the web server appears. Press **Enter** to accept the path shown, or enter a different path.
2. Select the output device to produce the journal. See "Reports" on page 1-29 for more information on output devices.

Transaction Journal Report

10/13/2005 3:16 PM			Builders Supply Transaction Journal	Page 1
Order No.	Date	Time	New Order No.	Additional Comments

00000142	10/10/2005	12:47 PM		Please process promptly.
00000144	10/10/2005	4:15 PM		
00000149	10/11/2005	10:09 AM		
00000150	10/13/2005	1:22 PM		Note shipping address change.
00000151	10/13/2005	2:39 PM		May substitute standard window.
End of report				

Build Sales Orders from Remote Files

You can elect to allow access to your OSAS data files directly from the Internet, or to copies of your files which have been placed on the web server. The process of using copies of your files is known as *remote processing*. See chapter 2 for more information about remote processing.

Use the **Build Sales Orders from Remote Files** function to copy orders entered through the web to quotes in your OSAS Sales Order data files.

Note

Do not use this function if you are not using remote processing with OSAS Web B2B. When you elect not to use remote processing, the orders are entered as quotes directly into your OSAS files.

Because remote processing uses a different set of Sales Order files, the order numbers may change when you copy the orders to the live system. This function reprints the Transaction Journal after the copy to show the new order numbers assigned during the copy process.

A sample of the Build Sales Orders from Remote Files Log is on page 5-9.

Build Sales Orders from Remote Files Screen

Build Sales Orders from Remote Files

Commands Edit Modes Other Help

Web Server Programs Path
C:/inetpub/wwwroot/osasweb/

Remote Data Files Path
D:/OSAS/data/

Destination Batch ID WEBORD

Company H 10/12/2005 Terminal T000 INS

1. The web server programs path you specified in the **Install Web Server Components** function appears. Press **Enter** to accept the path shown, or enter a different path.
2. The remote data files path you used in the **Install Web Server Components function** appears. Press **Enter** to accept the path shown, or enter a different path.
3. Enter the batch ID you want to use for the quotes that are created during the copy process.
4. Select the output device to produce the log. See “Reports” on page 1-29 for more information on output devices. After the orders have been copied and the log prints, the **Sales Order Processing** menu appears.

Inquiry

Build Sales Orders from Remote Files Log

10/15/2005			Builders Supply	Page 1
9:52 AM			Build Sales Orders from Remote Files	
Order No.	Date	Time	New Order No.	Additional Comments

00000142	10/10/2005	12:47 PM	00000157	Please process promptly.
00000144	10/10/2005	4:15 PM	00000158	
00000149	10/11/2005	10:09 AM	00000159	
00000150	10/13/2005	1:22 PM	00000160	Note shipping address change.
00000151	10/13/2005	2:39 PM	00000161	May substitute standard window.
End of report				

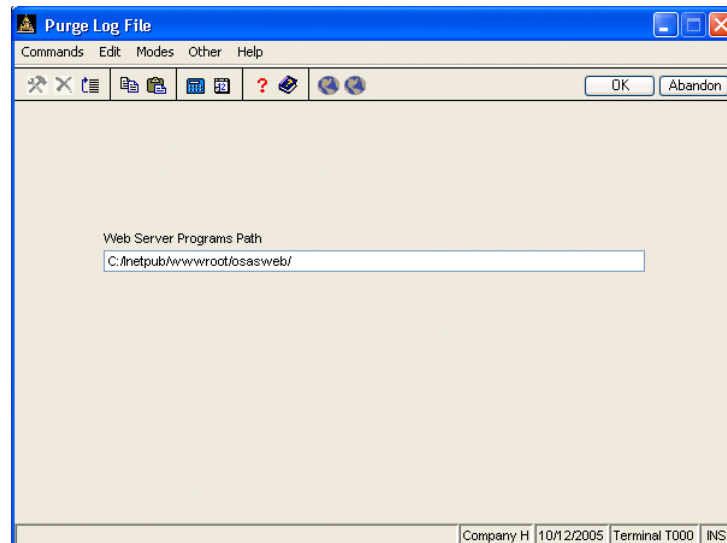
Purge Log File

Use the **Purge Log File** function to clear the log file for new orders. If you use remote processing, you should use this function after you use the **Build Sales Orders from Remote Files** function (page 5-7).

Note

You should print the Transaction Journal (page 5-5) before you run this function to preserve an audit trail of the orders that have been entered.

Purge Log File Screen



The web server programs path you entered in the **Install Web Server Components** function appears. Press **Enter** to accept this path, or enter a different one. Use the **Proceed (OK)** command to begin the purge process. When the purge finishes, the **Sales Order Processing** menu appears.

Remote Access

6

Remote Access	6-3
Copy Data Files to Web Server	6-5
Copy OSAS Programs to Web Server	6-7

Remote Access

When you set up OSAS Web B2B, you can elect to allow access to your OSAS data files directly from the Internet, or to allow access to copies of your files stored on the web server. The process of using copies of your files is known as *remote processing*. See chapter 2 for more information about remote processing.

Use the functions on the **Remote Access** menu to set up the programs and to refresh your data files if you use remote processing:

- Use the **Copy OSAS Programs to Web Server** function to set up remote processing.
- Use the **Copy Data Files to Web Server** function regularly to refresh the data stored on your web server. In addition, you should run this function every time you use the **Build Sales Orders from Remote Files** function on the **Sales Order Processing** menu.

Copy Data Files to Web Server

Use the **Copy Data Files to Web Server** function to refresh the data that is stored on the web server if you have chosen remote processing. This program copies the OSAS data files to the web server for the inquiry and sales order entry functions.

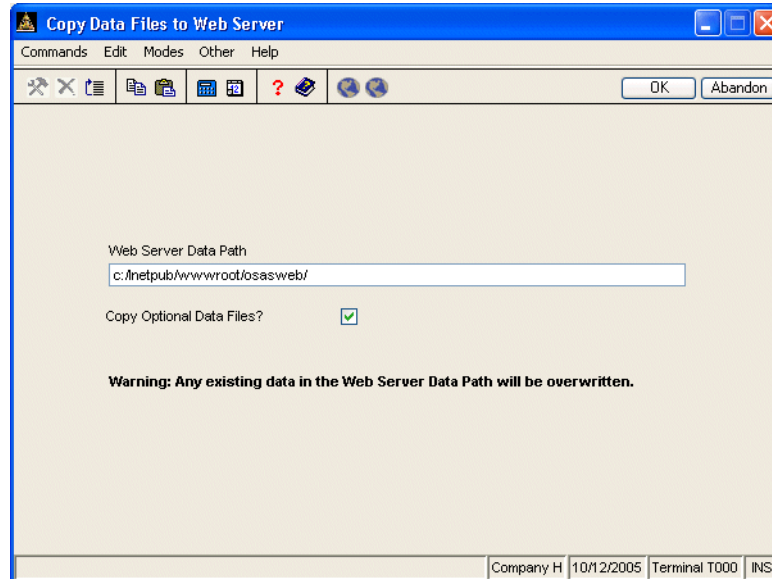
This function copies the Inventory, Accounts Receivable, and Sales Order files for use by OSAS Web B2B functions.

Note

You need to run this function only if you are using remote processing. If you are not using remote processing, your live OSAS data files are used for OSAS Web B2B functions.

You must copy the optional files to the web server at least once if you want your customers to be able to view history or inquire on invoices.

Copy Data Files to Web Server Screen



The web server data path you entered in the **Install Web Server Components** function (page 4-5) appears. Press **Enter** to use this path, or enter the full path from the OSAS system to the data path on the web server.

The AR Detail History file can be large and may take some time to copy. You can choose to skip copying this file to speed up the copy process. To copy the file, select the **Copy Optional Data Files** check box (or enter **Y** in text mode). To skip the file, clear this check box (or enter **N** in text mode).

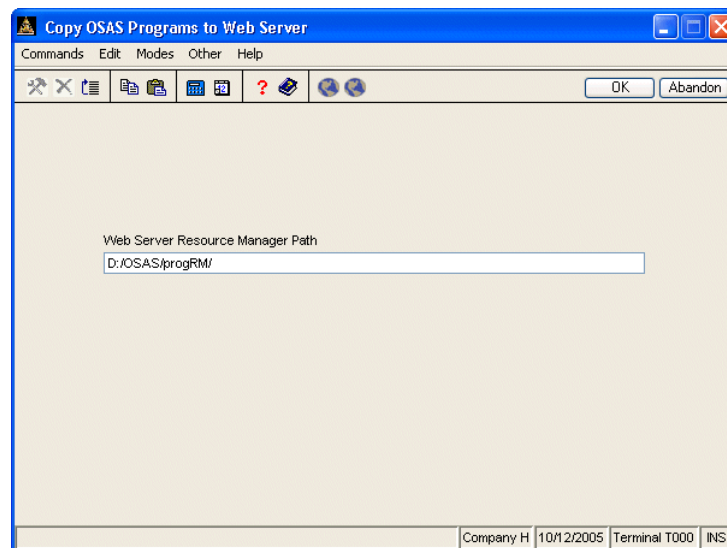
Use the **Proceed (OK)** command to begin the copy process. When copying finishes, the **Remote Access** menu appears.

Copy OSAS Programs to Web Server

Use the **Copy OSAS Programs to Web Server** function to set up the web server for remote processing. This program copies the OSAS program files to the web server for the inquiry and sales order entry functions.

You need run this function only during the installation process. Once the programs are copied to the web server, you do not need to copy them again, unless you change your web server or update the OSAS Web B2B software.

Copy OSAS Programs to Web Server Screen



The web server Resource Manager path you entered in the **Install Web Server Components** function (page 4-5) appears. Press **Enter** to use this path, or enter the full path from the OSAS system to the Resource Manager path on the web server.

Use the **Proceed (OK)** command to begin the copy process. When copying finishes, the **Remote Access** menu appears.

Master File Lists

7

Printing a Master File List	7-3
Item Group Codes List	7-5
Inventory Item Pictures List	7-7
Internet Companies List	7-9
Internet Inventory Item Groups List	7-11
Internet Access Codes List	7-13
Customer Groups List	7-15
Tables List	7-17

Printing a Master File List

The functions on the **Master File Lists** menu let you print lists of the information you entered using the **File Maintenance** menu. These lists do not contain any calculations or transaction amounts, but rather list only the basic file information used in the OSAS Web B2B system. If any of the information on a master file list is incorrect, use the appropriate function on the **File Maintenance** menu to correct it, then reprint the master file list.

You produce all master file lists the same way. Use the instructions below to print a master file list, modifying the procedure as necessary for the list you are printing. For example, if the screen for the list you want to print does not contain check box options, ignore that step and continue to the next.

Follow these steps to print a master list:

1. Select the list you want to print from the **Master File Lists** menu. The selection screen for that list appears. The Internet Inventory Item Groups List screen is shown below as an example.

Item ID	From	100
	Thru	920003
Location ID	From	CA0001
	Thru	TX0001
Group Code	From	APPLIANCES
	Thru	VMINDOWS

Print By:

Item ID
 Group Code

Company H | 10/12/2005 | Terminal T000 | INS

Inquiry

2. Select the range of information to include in the list in the **From** and **Thru** fields. The **Inquiry (F2)** command is usually available for these fields to let you select beginning and end range values from the list that appears.

Leave these fields blank to include all values in the list.

3. If the screen contains options that control the order in which information is printed (for example, print the list by item code or by group code), select the option you want to use to organize the information. You can select only one print option.
4. If the screen contains check boxes or Yes/No fields that control additional printing instructions, select the check box (or enter **Y** in text mode) to use that option when printing the list. Clear the check box (or enter **N** in text mode) if you do not want to use that option.
5. Select the output device to begin printing the list. See “Reports” on page 1-29 for more information. After you produce the list, the **Master File Lists** menu appears.

Item Group Codes List

The Item Group Codes List shows the item group codes that you set up using the **Item Group Codes** function on the **File Maintenance** menu.

Sample List

10/12/2005 3:54 PM	Builders Supply Item Group Codes List	Page 1
Group ID	Group Description	

APPLIANCES	Appliances	
DISHWASHER	Dishwashers	
DOORS	Doors	
HEAT/AIR	Heating and Air Conditioning Equipment	
INTERIOR	Interior Materials, Paint, Wallpaper, Supplies	
MATERIALS	Materials	
MILLWORK	Millwork Trim and Doors	
PAINT	Paints	
PARTS	Cabinet Replacement Parts	
REFRIG	Refrigerators	
STOVE	Stoves and Ranges	
SUPPLIES	Paint Supplies	
WINDOWS	Windows	
End of report		

Inventory Item Pictures List

The Inventory Item Pictures List shows the image files that you have associated with your inventory items for display on the web. You can select the range of items you want to include in the list.

Sample List

10/12/2005 3:54 PM	Builders Supply Inventory Item Pictures List	Page 1
Item ID	Item Description	Image File
100	Electrical Package	100.jpg
150	Plumbing Package	150.jpg
200200	Water Heater	200200.jpg
200300	Air Conditioner	200300.jpg
200400	Water Softener	200400.jpg
200500	Sump Pump	200500.jpg
200600	Humidifier	200600.jpg
900	Refrigerator - Black	900.jpg
901	Refrigerator - White	901.jpg

End of report

Internet Companies List

The Internet Companies List shows the companies that you have set up for Internet access for the companies you select.

Sample List

10/12/2005	Builders Supply	Page	1
3:55 PM	Internet Companies List		
Comp ID	Company Name	Batch	

H	Builders Supply	WEBORD	
End of report			

Internet Inventory Item Groups List

The Internet Inventory Item Groups List shows the item groups and the items that belong to them. The list can be printed for the items, locations, or group IDs you select, and can be sorted by item ID or by group code.

Sample List

Item ID	Item Description	Loc ID	Group Code	Image
10/12/2005 3:54 PM	Builders Supply Internet Inventory Item Groups List			Page 1
100	Electrical Package	CA0001	MATERIALS	100.jpg
100	Electrical Package	MD0001	MATERIALS	100.jpg
100	Electrical Package	MN0001	MATERIALS	100.jpg
100	Electrical Package	TX0001	MATERIALS	100.jpg
150	Plumbing Package	CA0001	MATERIALS	150.jpg
150	Plumbing Package	MD0001	MATERIALS	150.jpg
150	Plumbing Package	MN0001	MATERIALS	150.jpg
150	Plumbing Package	TX0001	MATERIALS	150.jpg
200	Heating/Cooling Package	CA0001	HEAT/AIR	
200	Heating/Cooling Package	MD0001	HEAT/AIR	
200	Heating/Cooling Package	MN0001	HEAT/AIR	
200	Heating/Cooling Package	TX0001	HEAT/AIR	
200100	Furnace	CA0001	HEAT/AIR	
200100	Furnace	MD0001	HEAT/AIR	
200100	Furnace	MN0001	HEAT/AIR	
200100	Furnace	TX0001	HEAT/AIR	
200200	Water Heater	CA0001	HEAT/AIR	200200.jpg
200200	Water Heater	MD0001	HEAT/AIR	200200.jpg
200200	Water Heater	MN0001	HEAT/AIR	200200.jpg
200200	Water Heater	TX0001	HEAT/AIR	200200.jpg
200300	Air Conditioner	CA0001	HEAT/AIR	200300.jpg
200300	Air Conditioner	MD0001	HEAT/AIR	200300.jpg
200300	Air Conditioner	MN0001	HEAT/AIR	200300.jpg
200300	Air Conditioner	TX0001	HEAT/AIR	200300.jpg
200400	Water Softener	CA0001	HEAT/AIR	200400.jpg
200400	Water Softener	MD0001	HEAT/AIR	200400.jpg
200400	Water Softener	MN0001	HEAT/AIR	200400.jpg
200400	Water Softener	TX0001	HEAT/AIR	200400.jpg

Internet Access Codes List

The Internet Access Codes List shows the access codes you set up for your customers for the companies, customers, or access codes you select.

Sample List

Company ID	Cust ID	Access Code	Location	HTML Page Type	Access

H	ACE001	Test1	MN0001	Information - Account Information	YES
				Orders - Order Inquiry	YES
				Orders - Order Inquiry - Detail Information	YES
				Information - History Inquiry	YES
				Inventory - Item Inquiry	YES
				Inventory - Item Inquiry - Show On Hand Quantity	YES
				Inventory - Item Inquiry - Show Price/Qty Breakdown	YES
				Information - Account Information - Show Balances	YES
				Information - Account Information - Show Credit Limit	YES
				Orders - Order Entry	YES
				Inventory - Item Inquiry - Additional Description	YES
				Information - Aged Trial balance	YES
				Information - Invoice Inquiry	YES
				Inventory - Item Inquiry - Link to Order Entry	YES
				Information - History Inquiry - Link to Order Entry	YES
				Orders - Order Inquiry - Link to Order Entry	YES
				Inventory - Item Inquiry - Show On Hand Quantity as 'Available'	NO
End of report					

Customer Groups List

The Customer Groups List shows the item groups that you have set up for each customer for the customer IDs or group codes you select. You can also choose to organize the list by customer or by group code.

Sample List

10/12/2005 3:55 PM	Builders Supply Customer Groups List	Page 1
Cust ID	Group ID	Group Description
ACE001	APPLIANCES	Appliances
ACE001	DOORS	Doors
ACE001	HEAT/AIR	Heating and Air Conditioning Equipment
ACE001	INTERIOR	Interior Materials, Paint, Wallpaper, Supplies
ACE001	MATERIALS	Materials
ACE001	MILLWORK	Millwork Trim and Doors
ACE001	PARTS	Cabinet Replacement Parts
ACE001	WINDOWS	Windows
CASHCA	INTERIOR	Interior Materials, Paint, Wallpaper, Supplies
CASHMD	INTERIOR	Interior Materials, Paint, Wallpaper, Supplies
CASHMN	INTERIOR	Interior Materials, Paint, Wallpaper, Supplies
CASHPS	INTERIOR	Interior Materials, Paint, Wallpaper, Supplies
CASHTX	INTERIOR	Interior Materials, Paint, Wallpaper, Supplies
DAL001	HEAT/AIR	Heating and Air Conditioning Equipment
DAL001	INTERIOR	Interior Materials, Paint, Wallpaper, Supplies
GRE001	HEAT/AIR	Heating and Air Conditioning Equipment
GRE001	INTERIOR	Interior Materials, Paint, Wallpaper, Supplies
KAN001	HEAT/AIR	Heating and Air Conditioning Equipment
KAN001	INTERIOR	Interior Materials, Paint, Wallpaper, Supplies
LOS001	HEAT/AIR	Heating and Air Conditioning Equipment
LOS001	INTERIOR	Interior Materials, Paint, Wallpaper, Supplies
SUN001	HEAT/AIR	Heating and Air Conditioning Equipment
SUN001	INTERIOR	Interior Materials, Paint, Wallpaper, Supplies
TEN001	HEAT/AIR	Heating and Air Conditioning Equipment
TEN001	INTERIOR	Interior Materials, Paint, Wallpaper, Supplies
VIS001	INTERIOR	Interior Materials, Paint, Wallpaper, Supplies

End of report

Tables List

The Tables List shows the tables that are on your system for the range of IDs you select.

Sample List

10/12/2005 3:55 PM	Builders Supply Tables List OSAS Web B2B	Page 1
Table ID COPYPATH	Description Web Path Table	
No. of Columns 1	Column Length 60	Type A
VALUE	-----	
C:/inetpub/wwwroot/osasweb/		
D:/OSAS/data/		
D:/OSAS/progRM/		
Table ID EMAIL	Description E-Mail Address in case of problems.	
No. of Columns 1	Column Length 50	Type A
Email Address	-----	
Webmaster@builders_supply.com		
Table ID HOMEPATH	Description Home Page Path Table	
No. of Columns 1	Column Length 60	Type A
VALUE	-----	
Builders' Supply		
C:/inetpub/wwwroot/		
98FB98		
000000		

The Web Interface

8

OSAS Web Login Page	8-3
Account Information	8-5
Aged Trial Balance	8-7
History Inquiry	8-9
Invoice Inquiry	8-11
Item Inquiry	8-13
Order Inquiry	8-19
Order Entry	8-21

OSAS Web Login Page

The OSAS web login page is the first of the web pages presented to Internet users. Your customers use this page to enter their customer ID and password, and to choose the company for which to view information or enter orders.

The login page is created by the **Create Login Page** function on the **Set Up Web Components** menu. See Chapter 4 for more information about this function.

OSAS Web Login Page HTML Screen



Enter the customer ID, password (or access code), and then select the company (if you gave that customer access to more than one company).

Click **Submit** to continue.

When you click **Submit**, OSAS Web B2B checks the customer ID and password and opens the Account Information web page. This page contains top and side menus based on the functions and information to which you have granted that customer access.

If the customer ID is not set up or the password is incorrect, an error message appears.

OSAS Web B2B Menus

There are three top level menu groups: Information, Inventory, and Orders. These top level menus contain one or more side menu functions, as follows:

- Information
 - Account Information
 - Aged Trial Balance
 - History Inquiry
 - Invoice Inquiry
- Inventory
 - Item Inquiry
- Orders
 - Order Inquiry
 - Order Entry

Each top level menu also contains a link that allows you to return to the Login page to log in using a different customer ID, password, or company.

Account Information

The Account Information web page on the **Information** menu displays data for the customer's account from the OSAS files. The information includes general information, and can optionally include aging balances and credit information.

You can specify the information available on this page using the **Customer Internet Access Codes** function on the **File Maintenance** menu.

Account Information HTML Screen

OSAS Customer Lookup - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Links

OPEN SYSTEMS INC.

Information Inventory Orders

Account Information
Aged Trial Balance
History Inquiry
Invoice Inquiry
Login Page

Customer ID	ACE001
Customer Name	ACE BUILDERS
Customer Phone Number	(505)555-1646
Credit Limit	\$315,000.00
Terms Code:	2PCT Discount 2%/10 Days, Net Due 30 Days

	Period to Date	Quarter to Date	Year to Date
Sales	\$15,785.91	\$205,896.44	\$1,654,466.26
Number of Invoices	2	4	14

Last Sale Number	12670074
Last Sale Date	12/21/2005

New Finance Charge	\$.00
Unpaid Finance Charge	\$.00
Current Due	\$101,270.45
Balance 31-60	\$.00
Balance 61-90	\$.00
Balance 91-120	\$.00
121+	\$9,028.76
Unapplied Credit	\$20,693.74-
Total Due	\$90,405.47

[Top of Page](#)

Problems or Questions
Email: Webmaster@buildersupply.com

This screen was produced through OSAS® Web B2B and OPEN SYSTEMS® Accounting Software from Open Systems, Inc. Copyright 1999,2001 Open Systems Holdings Corp. All rights reserved.

Internet

Click **Top of Page** to return to the top of the web page. When you finish reviewing the information, close the browser to end your session or choose a different menu selection from the side or top menus.

Aged Trial Balance

The Aged Trial Balance web page on the **Information** menu lists the balance and aged invoices for the customer's account from the OSAS **ARINx** file.

Aged Trial Balance HTML Screen

The screenshot shows a web browser window titled "OSAS Customer Aged Trial Balance - Microsoft Internet Explorer". The page features the OPEN SYSTEMS INC. logo and navigation tabs for "Information", "Inventory", and "Orders". A left sidebar contains links for "Account Information", "Aged Trial Balance", "History Inquiry", "Invoice Inquiry", and "Login Page".

Account Information:

Customer ID	ACE001
Customer Name	ACE BUILDERS
Customer Phone Number	(605)555-1646

Invoice Inquiry Table:

Invoice Number	Date	Status	Type	Amount	Current	31-60	61-90	91-120	Over 120	Unapplied
00002081	05/19/2004	Released	Invoice	\$9,728.76	\$0.00	\$0.00	\$0.00	\$0.00	\$9,728.76	\$0.00
09270023	05/19/2004	Released	Credit Memo	\$5,000.00-	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$5,000.00-
12670046	09/05/2005	Released	Invoice	\$74,619.56	\$74,619.56	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00
12670064	11/24/2005	Released	Payment	\$35,467.99-	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00
12670074	12/21/2005	Released	Invoice	\$21,722.07	\$21,722.07	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00
12670206	12/21/2004	Released	Invoice	\$100.00	\$0.00	\$0.00	\$0.00	\$0.00	\$100.00	\$0.00
12670207	12/21/2004	Released	Credit Memo	\$15,693.74-	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$15,693.74-
24889030	12/09/2005	Released	Invoice	\$4,928.82	\$4,928.82	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00

At the bottom of the page, there is a "Top of Page" link, contact information for "Problems or Questions" (E-mail: Webmaster@builders_supply.com), and a footer note: "This screen was produced through OSAS® Web B2B and OPEN SYSTEMS® Accounting Software from Open Systems, Inc. Copyright 1998,2001 Open Systems Holdings Corp. All rights reserved."

Click **Top of Page** to return to the top of the web page. When you finish reviewing the information, close the browser to end your session or choose a different menu selection from the side or top menus.

History Inquiry

The History Inquiry web page on the **Information** menu shows the invoice, credit memo, and payment history for this customer.

History Inquiry HTML Screen

The screenshot shows the 'OSAS Customer History Lookup' web page. The browser window title is 'OSAS Customer History Lookup - Microsoft Internet Explorer'. The page features the 'OPEN SYSTEMS INC.' logo and navigation tabs for 'Information', 'Inventory', and 'Orders'. The 'Information' tab is active, showing account details for 'ACE BUILDERS' with Customer ID 'ACE001' and Phone Number '605655-1646'. A 'History Inquiry' section contains a table of transactions:

Type	Invoice Number	Date	Item ID/Job ID	Description	Location ID	Quantity	Unit of Measure	Amount
Invoice	24889030	12/09/2005		Heating/Cooling Package	MN0001	1.0000	PKG	\$2,464.41
Payment	24889022	10/27/2005		PAYMENT RECEIVED		.0000		\$492,882.00
Invoice	24889022	09/03/2005		Heating/Cooling Package	MN0001	100.0000	PKG	\$246,441.00
Payment	24889014	01/27/2005		PAYMENT RECEIVED		.0000		\$49,288.20
Invoice	24889014	12/17/2004		Heating/Cooling Package	MN0001	10.0000	PKG	\$24,644.10
Payment	24889003	08/27/2004		PAYMENT RECEIVED		.0000		\$276,013.92
Invoice	24889003	07/11/2004		Heating/Cooling Package	MN0001	56.0000	PKG	\$138,006.96
Credit Memo	12670207	12/21/2005	600	Standard Window 24" X 40"	MN0001	3.0000	EA	\$859.95-
Credit Memo	12670207	12/21/2005	550	Millwork Package	MN0001	3.0000	PKG	\$4,303.77-
Credit Memo	12670207	12/21/2005	450	Slide by Window 24" x 40"	MN0001	6.0000	EA	\$2,269.67-

Below the table is a search box with the text 'Search for Invoice' and a 'Next' button. At the bottom left, there is a 'Top of Page' link and contact information: 'Problems or Questions E-mail: Webmaster@buildersupply.com'.

The page lists 10 history entries for this customer. If there are more entries after the ones listed, the **Next** button appears. If there are more entries before the ones listed, the **Previous** button appears. Click these buttons to scroll through the entries.

Click **Top of Page** to return to the top of the web page.

To find a specific invoice, enter the invoice number in the box and then click **Search For Invoice**.

You can also click on an item ID to create a line item for that item in your order entry shopping cart if you are permitted to do so based on your customer ID and password.

When you finish reviewing history, close the browser to end your session or choose a different menu selection from the side or top menus.

Invoice Inquiry

The Invoice Inquiry web page on the **Information** menu shows the invoices and credit memos in history for this customer. Click an invoice number to view more information about that invoice on the Invoice Detail web page (page 8-12).

Invoice Inquiry HTML Screen

OSAS Customer History Invoice Lookup - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Links

OPEN
SYSTEMS INC.

Information Inventory Orders

Account Information
Aged Trial Balance
History Inquiry
Invoice Inquiry
Login Page

Customer ID	ACE001
Customer Name	ACE BUILDERS
Customer Phone Number	(605)555-1646

Type	Invoice Number	Date
Invoice	12670074	12/21/2005
Invoice	24888090	12/09/2005
Invoice	12670064	11/24/2005
Invoice	12670054	10/17/2005
Credit Memo	09270023	09/27/2005
Invoice	12670046	09/05/2005
Invoice	24888022	09/03/2005
Invoice	12670041	08/09/2005
Invoice	12670038	07/18/2005
Invoice	12670035	06/21/2005
Invoice	12670033	05/03/2005
Invoice	12670030	04/03/2005
Invoice	12670027	03/02/2005
Invoice	12670025	02/16/2005
Invoice	12670021	01/26/2005
Invoice	12670206	12/21/2004
Credit Memo	12670207	12/21/2004
Invoice	24888014	12/17/2004
Invoice	12670018	12/03/2004
Invoice	12670016	11/03/2004
Invoice	12670015	10/04/2004
Invoice	12670013	09/02/2004
Invoice	12670010	08/17/2004
Invoice	12670001	07/14/2004
Invoice	24888003	07/11/2004

Click an invoice number in the list to view detail for that invoice. The Invoice Detail web page appears.

Click **Top of Page** to return to the top of the web page.

Next

Top of Page

Done Internet

Invoice Detail HTML Screen

When you click an invoice number on the Invoice Inquiry web page, OSAS Web B2B re-creates that invoice form from the entries in the AR and IN history files and displays the invoice's detail on this web page.

OSAS Customer History Invoice - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Refresh Print Mail Stop Links

OPEN SYSTEMS INC.

Information Inventory Orders

Account Information
Aged Trial Balance
History Inquiry
Invoice Inquiry
Login Page

Sold To: ATTN: ACCOUNTS PAYABLE **Ship To:** ACE BUILDERS
ACE BUILDERS 1588 SE 31ST STREET
1588 SE 31ST STREET PADUCAH KY 28655-7865
PADUCAH KY 28655-7865

Date	Rep ID	Order No.	Ord Date	Ship Via	Terms	Inv No.
10/17/2005	GPD	47590662			C.O.D.	12670054

Item/Description	Quantities	Units	Price	Amount
100 Electrical Package	Shipped 10.0000	PKG	\$360.7280	\$3607.28
150 Plumbing Package	Shipped 75.0000	PKG	\$952.9065	\$71467.99
450 Slide by Window 24" x 40"	Shipped 10.0000	EA	\$162.6770	\$1626.77
550 Millwork Package	Shipped 28.0000	PKG	\$1087.9050	\$30461.34
700 Cabinets	Shipped 45.0000	SET	\$1055.0924	\$47479.16

Location Tax Breakdown
Tax location not on file

Subtotal: \$154642.54

Non-Taxable	Taxable	Sales Tax	Freight	Misc	Invoice Total
\$.00	\$154642.54	\$.00	\$.00	\$.00	\$154642.54

Amount Paid: \$154642.54
Net Due: \$.00

[Top of Page](#)

Click **Top of Page** to return to the top of the web page. Close the browser to end the session or select an option from the top or side menus.

Item Inquiry

The Item Inquiry web page lists the inventory item groups to which this customer has access. The groups provide a convenient way to organize items for viewing.

Use the **Customer Groups** function (page 3-17) on the **File Maintenance** menu in OSAS to change the item groups that this customer can view.

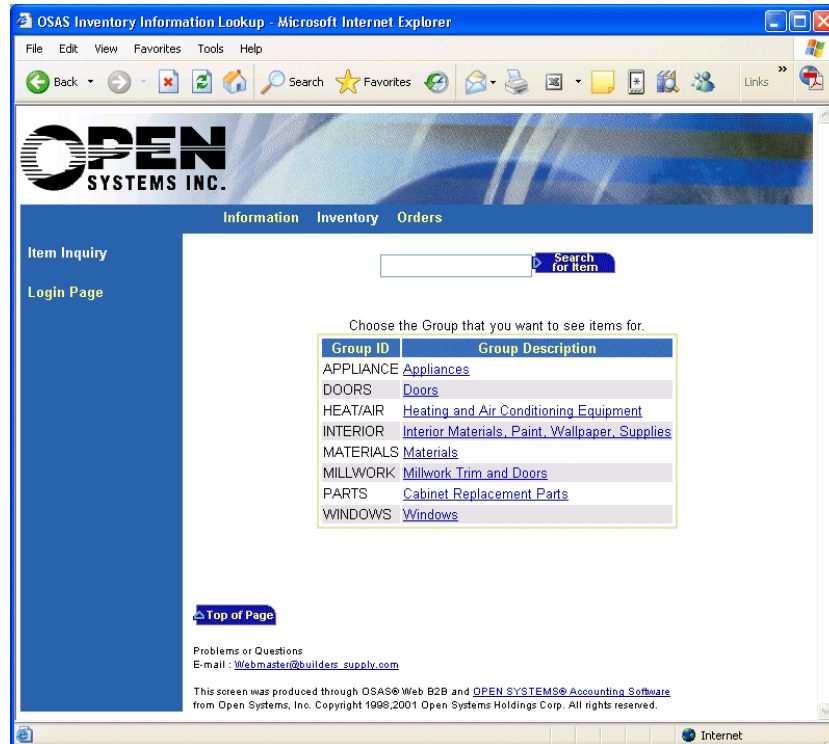
Use the **Internet Inventory Item Groups** function (page 3-9) on the **File Maintenance** menu in OSAS to change the items and subgroups in an item group.

Use the **Customer Internet Access Codes** function (page 3-13) on the **File Maintenance** menu in OSAS to choose whether to view quantities and price breaks on this web page.

Screen Use

Screen	Description
Item Group Codes	Use this screen to select the items you want to list.
Item Inquiry	This screen lists the items and subgroups in the group you select. The items appear in groups of 10.
Item Additional Descriptions	This screen lists the additional description lines for an item and a graphic image of the item, if you set one up.
Item Price Breaks	This screen shows the price breaks for an item you select (if you have access to this information).

Item Inquiry HTML Screen - Group Codes

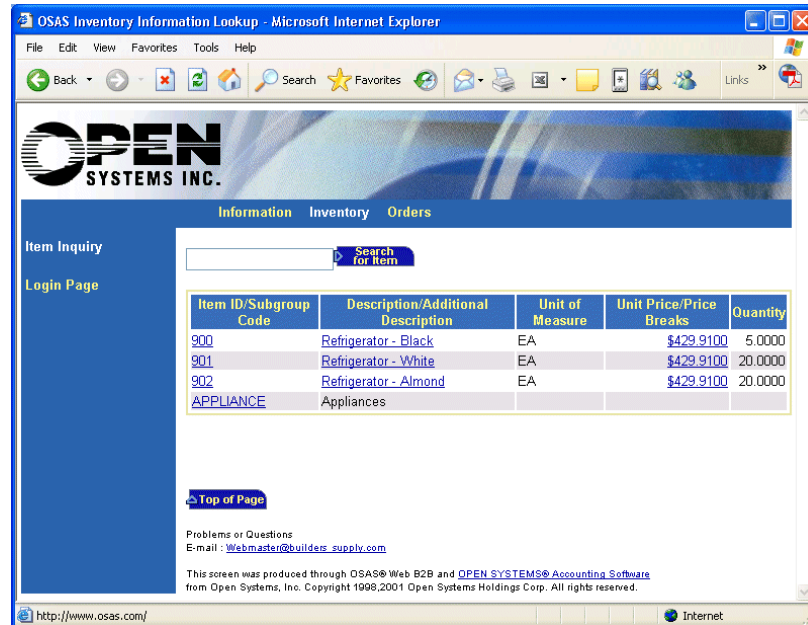


This screen lists 10 groups for this customer. If there are more groups after the ones listed, the **Next** button appears. If there are more groups before the ones listed, the **Previous** button appears. Click these buttons to scroll through the groups.

Click on the description of the item group you want to view. The screen changes to list the items and subgroups in that group.

If there are more than 10 item groups on file, enter the group code and click **Search for Group** to search for a specific item group. To search for a specific item within any group, enter the item ID and then click **Search for Item**.

Item Inquiry HTML Screen - Items



The items and subgroups that belong to the group you select appear. Units of measure, unit prices, and quantities appear for items in the group.

- If you click a subgroup ID, the screen changes to list the items in that group.
- If you click on an item ID, you add that item to your order entry shopping cart if you are permitted to do so based on your customer ID and password.
- If you click on a highlighted description, you see additional description information and a graphical image of the item, if available.
- If you click on the price of any item, you see a breakdown of quantity-break pricing for this customer.

The page lists 10 items and subgroups for this customer. If there are more than 10 items on file, the **Next** and **Previous** buttons appear. Click these buttons to scroll through the list of items.

To search for a specific item within any available group, enter the item ID and then click **Search for Item**.

Click **Top of Page** to return to the top of the web page.

When you finish reviewing the information, close the browser to end your session or choose a different menu selection from the side or top menus.

Item Additional Descriptions Screen

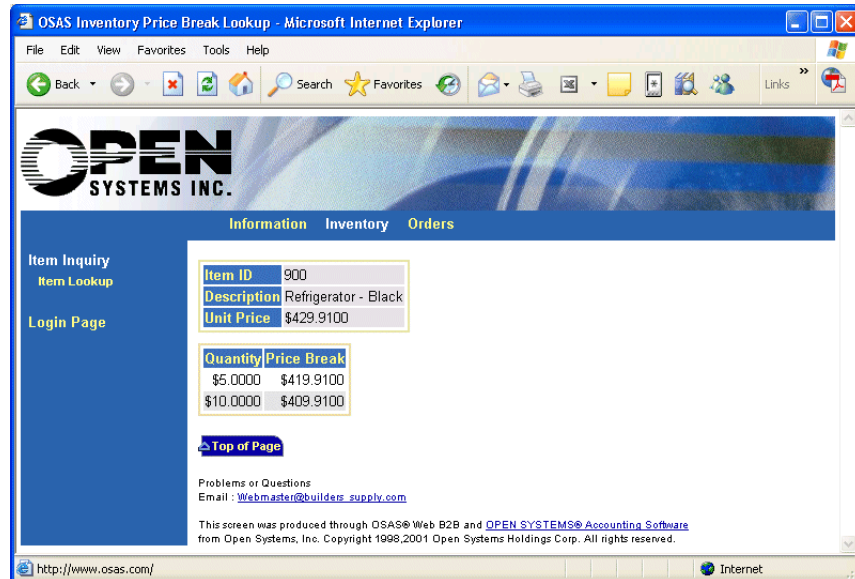


Click **Top of Page** to return to the top of the web page.

When you finish viewing additional descriptions, use your browser's **Back** button to return to the Item Inquiry web page.

When you finish reviewing items, close the browser to end your session or choose a different menu selection from the side or top menus.

Item Price Breaks HTML Screen



Click **Top of Page** to return to the top of the web page.

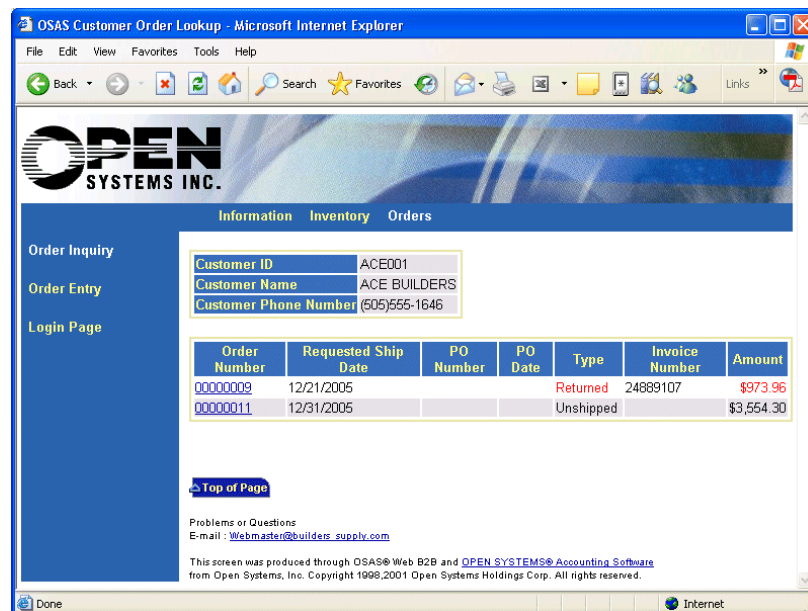
When you finish viewing price and price break information, use your browser's **Back** button to return to the Item Inquiry web page.

When you finish reviewing items, close the browser to end your session or choose a different menu selection from the side or top menus.

Order Inquiry

The Order Inquiry web pages display the orders on file from a customer. The information includes the shipping status of the order, and, optionally, the line item detail for individual orders.

Order Inquiry HTML Screen



The screen lists up to 10 orders for this customer. Returned orders appear in red. If there are more than 10 orders for this customer, the **Next** and **Previous** buttons appear. Click these buttons to scroll through the order numbers.

If you have access to Order Inquiry detail information, you can view it by clicking on any highlighted order number. The screen changes to list detail for that order number. When you finish viewing order information, close the browser to end your session or choose a different menu selection from the side or top menus.

Order Detail HTML Screen

OSAS Customer Order Detail Lookup - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites Links

OPEN SYSTEMS INC.

Information Inventory Orders

Order Inquiry
Order Entry
Login Page

Customer ID: ACE001
Customer Name: ACE BUILDERS
Customer Phone Number: (505)555-1646

Line Number	Item ID	Item Location	Item Description	Ordered Qty	Shipped Qty	Backordered Qty	Unit of Measure	Extended Price
001	100	CA0001	Electrical Package	1.0000	.0000	.0000	PKG	\$380.55
002	150	CA0001	Plumbing Package	3.0000	.0000	.0000	PKG	\$3,173.75

[Top of Page](#)

Problems or Questions
E-mail: Webmaster@builders_supply.com

This screen was produced through OSAS® Web B2B and OPEN SYSTEMS® Accounting Software from Open Systems, Inc. Copyright 1998,2001 Open Systems Holdings Corp. All rights reserved.

Done Internet

This page lists up to 10 line items for the order selected. If there are more line items for the order, the **Next** and **Previous** buttons appear. Click these buttons to scroll through the line items for the order.

Click **Top of Page** to return to the top of the web page.

When you finish viewing line item detail, use your browser's **Back** button to return to the Order Inquiry web page. When you finish viewing order information, close the browser to end your session or choose a different menu selection from the side or top menus.

Order Entry

The Order Entry web pages allow your customer to enter a sales order through the Internet.

The customer chooses the items he wants from the item groups to which you grant him access. Next, the customer enters the quantity of the item that he wants to purchase and adds it to his shopping cart. The OSAS Web B2B system then calculates the price and tax for the items the customers ordered, and lists the order total.

When the customer submits the order and verifies the shipping information, OSAS Web B2B creates the order as a quote using the batch ID you specified in the Resource Manager **Company Information** function in OSAS.

Screen Use

Screen	Description
Order Entry Group Codes	Use this screen to select the item group that contains the items you want to order.
Order Entry Items	Use this screen to select the item you want to order from the group you selected.
Order Entry Line Item	Use this screen to enter the quantity of the item you want, to calculate the price, and to add the item to your shopping cart.
View Shopping Cart	Use this screen to review your order, to change the quantities on any item, and to submit your order.
Shipping Information	Use this screen to verify the shipping address for this order and to enter special notes about it.
Order Number	This page displays the number that identifies your order.

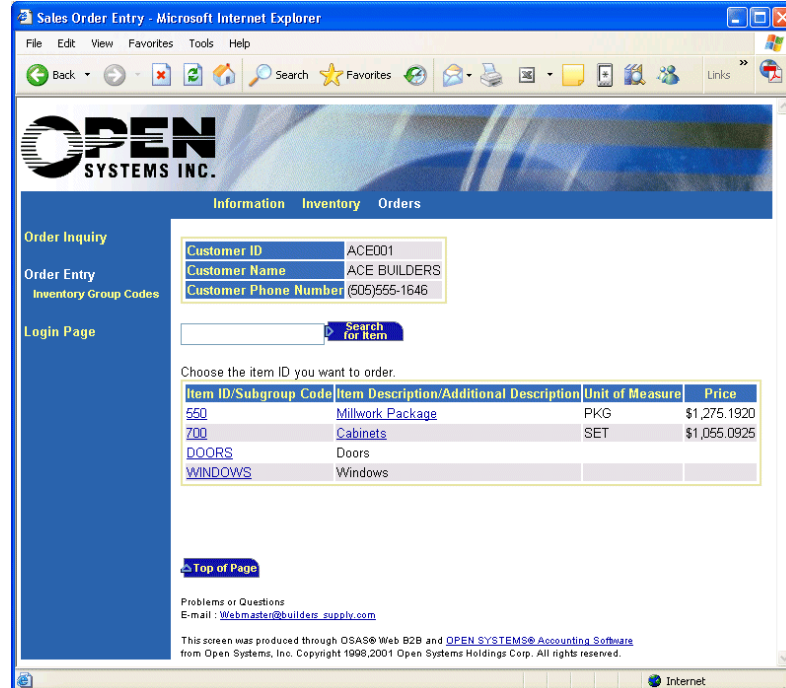
Order Entry Group Codes HTML Screen



The page lists up to 10 groups for this customer. If there are more groups after the ones listed, the **Next** and **Previous** buttons appear. Click these buttons to scroll through the list of groups.

Click on the item group for which you want to list items, or (if there are more than 10 groups available to you) enter a group to search for and click the **Search for Group** button. You can also search for a specific item from all available groups by entering the item ID and clicking the **Search for Item** button. If there are items in your shopping cart, the **View Shopping Cart** command appears in the side menu. Click **View Shopping Cart** to list its contents.

Order Entry Items HTML Screen



The items and subgroups that belong to the group you select appear. Units of measure and unit prices appear for items in the group.

- When you click on a subgroup ID, you see the items in that group.
- When you click on an item ID, the Order Entry Line Item page appears.
- When you click on a highlighted description, you see additional description information and an image of the item, if available (see page 8-17 for a sample of the additional descriptions page).

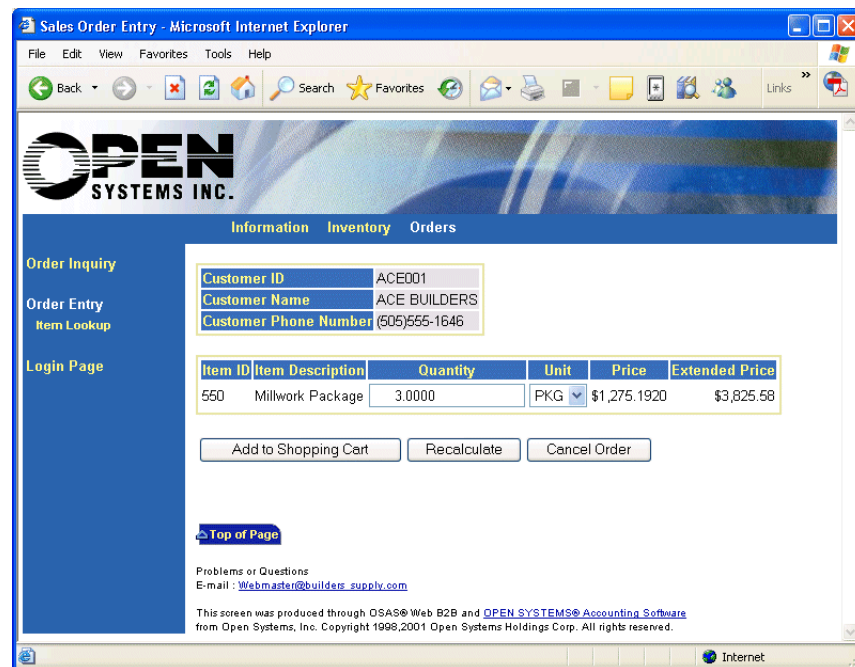
The screen lists up to 10 items and subgroups for the item group. If there are more than 10 items, the **Next** and **Previous** buttons appear. Click these buttons to scroll through the items and subgroups.

You can also search for a specific item within any available group by entering the item ID and clicking **Search for Item**.

If there are items in your shopping cart, the **View Shopping Cart** command appears in the side menu. Click **View Shopping Cart** to list its contents.

If you close your browser or leave the Order Entry web pages, your shopping cart remains intact for 24 hours. When you return to the site within the allotted time period, you can resume your order entry from where you left off.

Order Entry Line Item HTML Screen



Customer ID: ACE001
 Customer Name: ACE BUILDERS
 Customer Phone Number: (505)555-1646

Item ID	Item Description	Quantity	Unit	Price	Extended Price
550	Millwork Package	3.0000	PKG	\$1,275.1920	\$3,825.58

Buttons: Add to Shopping Cart, Recalculate, Cancel Order

Top of Page

Problems or Questions
 E-mail: Webmaster@builders_supply.com

This screen was produced through OSAS® Web B2B and OPEN SYSTEMS® Accounting Software from Open Systems, Inc. Copyright 1998,2001 Open Systems Holdings Corp. All rights reserved.

Enter the quantity of the item you want to order and select the appropriate unit from the **Unit** list box. Enter **0** (zero) if you do not want to order this item.

After you specify the quantity you want, click **Recalculate**. OSAS Web B2B calculates the extended price for the quantity you ordered.

When you are satisfied with the quantity and price, click **Add to Shopping Cart** to add the item to your cart. The Order Entry Items screen appears.

To abandon your entire order and empty your shopping cart, click **Cancel Order**. The Order Entry Group Codes page appears.

If there are items in your shopping cart, the **View Shopping Cart** command appears in the side menu. **Click View Shopping Cart** to list its contents so that you can review your order or check out.

If you close your browser or leave the Order Entry web pages, your shopping cart remains intact for 24 hours. When you return to the site within the allotted time period, you can resume your order entry from where you left off.

View Shopping Cart HTML Screen

Customer ID: ACE001
 Customer Name: ACE BUILDERS
 Customer Phone Number: (605)555-1646

To remove an item change the quantity to zero.

Item ID	Item Description	Quantity	Unit	Price	Extended Price
550	Millwork Package	3.0000	PKG	\$1,275.1920	\$3,825.58
250	Exterior Panels	1.0000	CS	\$1,410.0135	\$1,410.01
650	Steel Supports	1.0000	PKG	\$14,852.6765	\$14,852.68
Tax					\$0.00
Total Price					\$19,888.27

Check Out Recalculate Cancel Order Change Order to Taxable

The screen lists up to 10 line items for this order. If there are more than 10 items, the **Next** and **Previous** buttons appear. Click these buttons to scroll through the list of line items.

Review the items you ordered. You can change the quantity of any line item, or cancel a line item by changing the quantity to zero. If you change the quantity of a line item, click **Recalculate** to recalculate the order totals based on your changes.

To abandon your entire order and empty your shopping cart, click **Cancel Order**. The Order Entry Group Codes page appears.

If you are a tax-exempt customer, but you must pay tax on this order, click **Change Order to Taxable** to calculate the sales tax for the order.

When you are satisfied with your order, click **Check Out**. The Shipping Information page appears.

If you close your browser or leave the Order Entry web pages, your shopping cart remains intact for 24 hours. When you return to the site within the allotted time period, you can resume your order entry from where you left off.

Select Shipping Address HTML Screen

OPEN SYSTEMS INC.

Information Inventory Orders

Order Inquiry
Order Entry
Item Lookup
View Shopping Cart
Login Page

Bill To:

Customer ID	ACE001
Customer Name	ACE BUILDERS
Address	1588 SE 31ST STREET
City	PADUCAH
State	KY
Zip	28655-7865
Country	US

Ship To: Billing Address

[Top of Page](#)

Problems or Questions
E-mail : Webmaster@builders-supply.com

This screen was produced through OSAS® Web B2B and OPEN SYSTEMS® Accounting Software from Open Systems, Inc. Copyright 1998,2001 Open Systems Holdings Corp. All rights reserved.

The billing and ship-to addresses you set up for the customer in the Accounts Receivable **Customers** and **Ship-to Addresses** appear in the **Ship To** list box (this list is limited by available memory). If you have not set up any ship-to addresses, only the customer's billing address appears in the list and is used as the default address.

Select the address to which to ship goods. Click **Submit** to proceed to the Shipping Information page where you can verify and change shipping address data.

Shipping Address Information HTML Screen

OPEN SYSTEMS INC.

Information Inventory Orders

Order Inquiry
Order Entry
Item Lookup
View Shopping Cart
Login Page

Customer ID	ACE001	Ship To:	
Customer Name	ACE BUILDERS	Name	ACE BUILDERS
Address	1588 SE 31ST STREET	Address	1588 SE 31ST STREET
City	PADUCAH	City	PADUCAH
State	KY	State	KY
Zip	28655-7865	Zip	28655-7865
Country	US	Country	U.S.A.

Ship Method	Federal Exp 2nd-Day	Month	Day	Year	
Your PO Number	115469	Your PO Date (MM/DD/YYYY)	10	13	2005
		Requested Ship Date (MM/DD/YYYY)	10	20	2005

Additional Comments

[Submit](#)

[Top of Page](#)

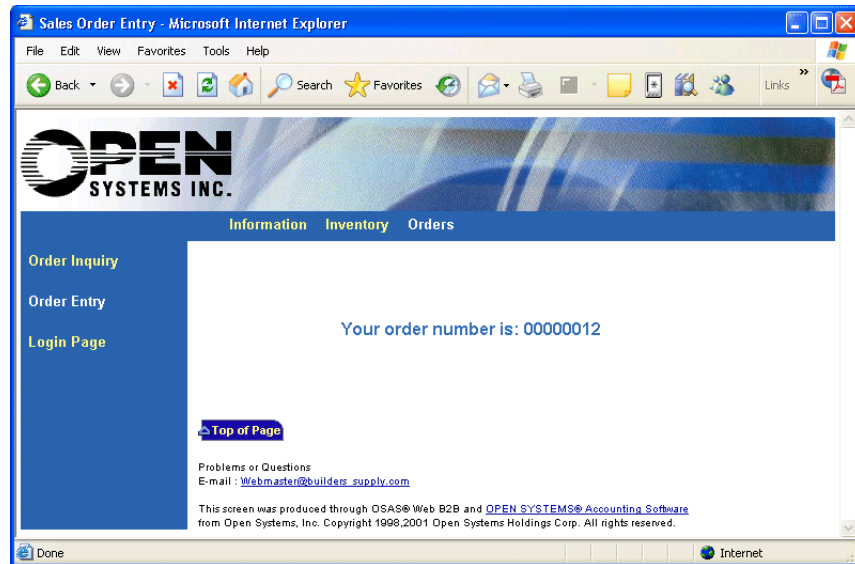
Make corrections to the shipping information as necessary. Refer to the field definitions for descriptions of specific fields. When you finish, click **Submit** to process your order. After your order is submitted successfully, the Order Number Web Page appears.

Field Definitions

Field Name	Description
Name	Enter a company's or individual's name for the shipment.
Address	Enter 1-3 lines of address information for the shipment.
City/State/ Zip/Country	Enter the city, state, zip code, and country to which the shipment will be sent.
Ship Method	Select a shipping method from the list presented.
Your P.O. Number	Enter your purchase order number, if necessary.
Your P.O. Date	Enter the date of your purchase order. Enter the month, day, and year in the appropriate fields.
Requested Ship Date	Enter the date by which you want to receive the shipment. Enter the month, day, and year in the appropriate fields.
Additional Comments	Enter any additional information you want to pass along about your order. You might use this field to specify alternative items, to request a rush order, to request a phone call before the order is shipped, and so on.

If you close your browser or leave the Order Entry web pages, your shopping cart remains intact for 24 hours. When you return to the site within the allotted time period, you can resume your order entry from where you left off.

Order Number HTML Screen



The number assigned to the order appears. After you record the order number, you can close the browser to end your session, or choose a different menu selection from the side or top menus.

Note

If you use remote processing, this number may change when you run the **Build Sales Orders from Remote Files** function (page 5-7) on the **Sales Order Processing** menu.

System Messages

A

Access code {access code} for company {company ID} and customer {customer ID} does not exist.

The access code from which you are trying to copy is not set up for the customer and company entered. Check the access code and verify that it is set up correctly with the customer and company from which you are trying to copy.

Company is not set up for Internet Transactions.

You cannot use the **Build Sales Orders from Internet** function for a company listed as **Inactive**. Use the **Company Information** function on the Resource Manager **Company Setup** menu to change the **OSAS Web** status to **Active**.

Could not find {file name}. You may have problems running OSAS Web without this file.

A required file is missing. Without this file, some functions within OSAS Web B2B will not work properly. Make sure the required file is in the data path and try again.

Could not find {file name}. You may have problems running OSAS Web without this program.

A required program is missing. Without this program, some functions within OSAS Web B2B will not work properly. Make sure the required program is in the Resource Manager path and try again.

Could not find application information for Resource Manager.

The file **OSAPPL** is missing or corrupted. Make certain that **OSAPPL** is in your Resource Manager path and try the function again.

System Messages

Customer {customer ID} already has internet access.

The customer is already set up within this group.

Destination directory cannot be the same as source directory.

The destination directory entered is the same as the source directory. Check for the correct destination directory and reenter it.

Internet access for {customer ID} in company {company ID} will be completely removed.

Use the **Delete (F3)** command to remove all access codes for the customer and the company or press **Enter** to abort this operation.

Internet access code {access code} will be removed.

Use the **Delete (F3)** command to remove the access code for the customer or press **Enter** to abort this operation.

Inventory must be installed to run the Print Inventory Item Groups list.

The OSAS Inventory application must be installed to use this function.

Inventory must be installed to run the Print Inventory Pictures list.

The OSAS Inventory application must be installed to use this function.

Item {item ID} is already set up.

The item ID you entered already has Internet access. Either enter a new item or edit the existing item entry.

Line Item error: &, ?, #, +, or “, these are reserved for internet use.

The code or ID you entered contains an invalid character. Reenter the code or ID without these reserved characters.

Master access code {access code} for company xxx does not exist.

The access code you entered is not set up for the company that you entered. Check the code to verify that it is set up correctly for the company from which you are trying to copy.

The file name cannot contain spaces.

The file name you entered for a graphic cannot have a space in it. Check the name of the picture file and reenter it.

Thru value must be greater than from value.

When you specify a range of items for a report, the **Thru** value you enter must be greater than or equal to the **From** value.

The code you want to use for a master password is already being used.

The master password you entered is already set up for one of your customers. Choose another password.

Access code {access code} is a master password.

The password you are entering for the customer is already set up as a master password. Choose another password.

There are no internet companies set up to make a login page for.

You must flag at least one company for Internet access using the **Company Information** function in Resource Manager before you can create your login page.

The last character of your path must be a '/

The path entered is too long. The path name must end in a '/

System Messages

Unable to:

open data directory for the Web Server.
locate OWORDLOG log file.
open specified directory.
find SOTHxxx on Web Server Path.
find SOTDxxx on Web Server Path.
open specified directory.
open specified path for your home page.
open osasweb directory on the Web server.
open BBx directory to the server.
open osasweb directory on the Web Server.

The system was unable to find the path you entered. Find the correct path name and enter it.

You must have Sales Order installed to use Build Sales Orders.

The OSAS Sales Order application must be installed to use this function.

You must set up the data directory on the web server.

When you used the **Install Web Server Components** function, you did not enter the **Data File Path**. You must set up this path before you can use the **Build Sale Orders from Internet** function.

You cannot delete a group that has items set up for it.

The group code you are trying to delete has items set up for it. Use the **Internet Inventory Item Groups** function on the **File Maintenance** menu to delete all the items from this group.

You must have Inventory installed for this company to use item groups.

The OSAS Inventory application must be installed to use this function.

You must set up internet companies before setting up internet items.

The company for which you are trying to set up items does not have Internet access. Set up Internet access for the company before using this function.

You must have Inventory installed for this company to use pictures.

The OSAS Inventory application must be installed to use this function.

You cannot delete a group that is currently set up under a customer.

The group code you are trying to delete has customers set up for it. Use the **Customer Groups** function to delete all the customers from this group.

You must run 'Install OSAS Web Components' first.

You have not run the **Install Web Server Component Function** from the **Set Up Web Components** menu. You must run this function before you can use any function on the **Sales Order Processing** menu.

You must set up the osasweb directory on the server first.

When you used the **Install Web Server Components**, you did not enter the **Web Server Programs Path**. This path needs to be set up before you can use this function.

You must set up internet companies before setting up customer groups.

The company for which you are trying to set up customer groups does not have Internet access. Set up Internet access for the company before using this function.

You must set up group codes before setting up customer groups.

The company you are trying to set up customer groups for does not have item group codes set up. Setup item group codes for the company before using this function.

You must have Accounts Receivable installed to set up customer groups.

The OSAS Accounts Receivable application must be installed to use this function.

System Messages

You must have Sales Order installed to view the transaction journal.

The OSAS Sales Order application must be installed to use this function.

You must have Accounts Receivable installed to set up this Company

The OSAS Accounts Receivable application must be installed to use this function.

You must have Sales Order installed to use purge log file.

The OSAS Sales Order application must be installed to use this function.

You must set up the BBx directory to the server first.

You did not specify the **Web Server Programs Path** using the **Install Web Server Components** function. This path must be set up before you can use this function.

Common Questions

B

When I try to access the web, I get an error. Why?

A drive in your computer is mapped but not used by OSAS Web B2B and should be excluded in the CONFIG.BBX file. Add this line to your CONFIG.BBX file (specifying the appropriate drive letter):

dsksyn E:

Specify a **dsksyn** line for each drive mapped but not used by OSAS Web B2B. Read Appendix D for more help on editing your CONFIG.BBX file.

How many users can access OSAS Web B2B at the same time?

The number of customers OSAS Web B2B allows to access your data depends on two things: your web server and the BBx interpreter. While the number of users depends on the program's limitations, you can control the number of users within certain parameters.

The web server can be set to allow only a limited number of users, depending on the server's capabilities. Consult your web server's guide for more help.

Your BBx interpreter can handle a given number of users, depending on your activation key. If more than that number of users try to access functions in OSAS Web B2B at the same time, the users that logged on last will receive an error.

Note

An interpreter session is only open for as long as it takes to load the HTML page. When the page is finished loading, the session closes, allowing another user to have access.

Common Questions

I set up item pictures and they don't show up. Why not?

While setting up your item pictures, you must set up the virtual paths linking the data to the display function screens. Check these two items:

1. You must set up a virtual path on the web server. The virtual path should be **/image**, with the actual path going to the directory where the images are stored. See your web server manual for more information.
2. You need to copy the image files into the directory on the web server to which the virtual path is mapped.

Can I use WIN CGI scripting calls with OSAS Web B2B?

No. Currently OSAS Web B2B only supports standard CGI scripting calls.

I run OSAS Web B2B at my Internet Solution Provider (ISP). How do I update my files?

Follow these steps:

1. Find out what directories the ISP is using for your OSAS Web B2B files. The directories you need to check are:
 - The actual path to which the **/osasweb** virtual directory is mapped.
 - The actual path to which the **/image** virtual directory is mapped.
 - The actual path of the document directory in which **OPENHOME.HTM** is located.
 - The directory where the data and OSAS programs are stored (if you do not store them in the **/osasweb** virtual directory).

2. Create a *mirror web site*. A mirror web site is a copy of your web site, including file placement, that you may create on a separate computer at your location. Because it matches your ISP web site, you can use the mirror web site to create what your customers see when you personalize your site. This saves the potential down time that occurs when you test various OSAS Web B2B functions, solve problems, and make repeated web site modifications without constantly updating your ISP web site. When you update your files and functions, make your modifications to your mirror web site, check them, and then copy the updates to your ISP web site.
3. To update your login page, use the **Create Login Page** function and enter the path for the document directory on the mirror web site.
4. To update all the OSAS Web B2B programs, use the **Install OSAS Web Components** function, and set the paths to use the directories on the mirror web site.
5. To update the remote data files and remote programs, use the **Copy Data Files to Data Server** and **Copy OSAS Programs to Web Server** functions. After completing these functions, set the ISP path to match the mirror web site.
6. To update the image files, copy them into the image directory on the mirror web site.
7. When you finish updating your files on the mirror web site, you can use FTP (file transfer protocol) software to move all the files in each directory to the matching ISP directory.

I'm using .BMP file formats for my pictures and they don't display correctly.

Many Internet browsers do not support the viewing of bitmap (.BMP) images. You should try using .JPG or .GIF file formats instead. Files saved in these formats are better quality, show much more detail and, in addition to working better on browsers, load faster than a bitmap file.

List of Files Copied

C

When you install the web server components, the following files are copied from the OSAS progOW path to the destination you entered. Some files are stored with different extensions to avoid duplication. These files are renamed to the correct extension when they are copied.

BBX Programs

These programs use input from users to look up data and create web pages:

OWPASSWD.BBX	Verifies passwords
OWMENU.BBX	Initialization program for the session
OWARMENU.BBX	Creates the Information menu
OWINMENU.BBX	Creates the Inventory menu
OWSOMENU.BBX	Creates the Orders menu
OWMNUBLD.PUB	Builds the side and top navigation menus
OWLOOKUP.BBX	Calls a BBx program based on your choice from the main menu
OWCUST.BBX	Information—Account Information
OWAGE.BBX	Information—Aged Trial Balance
OWINV.BBX	Information—Invoice Inquiry
OWINVPR1.BBX	Information—Invoice Inquiry Detail
OWORDR.BBX	Order Inquiry—Information
OWORD2.BBX	Order Inquiry—Detail Information

List of Files Copied

OWORD6.BBX	Order Entry—Select Shipping Address
OWHIST.BBX	Information—Detail History Inquiry
OWINVL.BBX	Item Inquiry—Group Code Breakdown
OWITEMS.BBX	Item Inquiry—Items for the group code you selected
OWPRCALC.BBX	Item price calculator
OWPBRK.BBX	Item Inquiry—Price Break Information
OWADDL.BBX	Additional Description/Picture
OWORDER.BBX	Order Entry—Group Code Screen
OWORDER2.BBX	Order Entry—Items for the group code you selected
OWORDER3.BBX	Order Entry—Line Item Entry
OWORDER4.BBX	Order Entry—Recalculate Prices and Submit Line
OWORDER5.BBX	Order Entry—View/Submit Order
OWORDER6.BBX	Order Entry—Recalculate Prices and/or Show Shipping Information for submitting an order
OWORDER7.BBX	Order Entry—Move an order into Sales Order
OWPATH.BBX	Get the path of the BBx directory on the web server
OWSESS.PUB	Verify the SESSION ID
OWFILE.PUB	Set the template for the SESSION string
OWPURGE.PUB	Remove all entries from the SESSION file that are more than 24 hours old
OWPC.BBX	Get period information from RMTB
UTHTMCON	Builds conditional hypertext links based on input provided from the calling program
UTHTMFIL.BBX	Builds odd/even row coloration based on the stylesheet

HTML Base Pages

These files contain the framework for the HTML pages displayed to Internet users. The data requested by the users is merged into the framework by the programs listed above.

OWPASSWD.TXT	Template text file for OWPASSWD.BBX
OWMENU.TXT	Template text file for OWMENU.BBX
OWBASE.TXT	Template text file for elements common to all screens.
OWCUST.TXT	Used by OWLOOKUP
OWAGE.TXT	Used by OWAGE.BBX
OWINV.TXT	Template text file for OWINV.BBX
OWINVPR.TXT	Template text file for OWINVPR.BBX
OWORDR.TXT	Used by OWLOOKUP
OWORD2.TXT	Template text file for OWORD2.BBX
OWORD6.TXT	Template text file for OWORD6.BBX
OWHIST.TXT	Used by OWLOOKUP
OWINVL.TXT	Used by OWLOOKUP
OWPBRK.TXT	Template text file for OWPBRK.BBX
OWSESS.TXT	Invalid Session message
OWORDASK.TXT	Continue/New Order message
OWORDER.TXT	Template text file for OWORDER.BBX
OWORDER2.TXT	Template text file for OWORDER2.BBX
OWORDER3.TXT	Template text file for OWORDER3.BBX
OWORDER5.TXT	Template text file for OWORDER5.BBX
OWORDER6.TXT	Template text file for OWORDER6.BBX
OWORDER7.TXT	Template text file for OWORDER7.BBX
OWBADQTY.TXT	Invalid Quantity message

List of Files Copied

OWADDL.TXT	Template text file for OWPASSWD.BBX
OWITEMS.TXT	Used by OWLOOKUP
OWBUSY.TXT	Process Interrupted message
OWNOVIEW.TXT	No Items to View message

Standard CGI Executables

These programs accept user input, start the BBx interpreter, and run the necessary OSAS Web B2B program. When the BBx program is finished, these programs load the resulting web pages:

OWMENU.EXE	OWORD2.EXE	OWORDER3.EXE
OWARMENU.EXE	OWORD6.EXE	OWORDER4.EXE
OWINMENU.EXE	OWPBRK.EXE	OWORDER5.EXE
OWSOMENU.EXE	OWADDL.EXE	OWORDER6.EXE
OWLOOKUP.EXE	OWORDER.EXE	OWORDER7.EXE
OWINVPR.EXE	OWORDER2.EXE	

Files Copied and Function in OSAS Web B2B

When you copy files from OSAS Web B2B, files are automatically created, renamed, and copied to in their destination drive. The tables below shows the original file name, the temporary file that is created and the final file name.

For a UNIX/Linux-based system, the files are:

Original File	Temporary File	Destination File
owmenu.tm2	Builds owmenu.sh	owmenu.exe
owarmenu.tm2	Builds owarmenu.sh	owarmenu.exe
owinmenu.tm2	Builds owinmenu.sh	owinmenu.exe
owsomenu.tm2	Builds owsomenu.sh	owsomenu.exe
owlookup.tm2	Builds owlookup.sh	owlookup.exe

List of Files Copied

oword2.tm2	Builds oword2.sh	oword2.exe
owpbrk.tm2	Builds owpbrk.sh	owpbrk.exe
owaddl.tm2	Builds owaddl.sh	owaddl.ex
oworder.tm2	Builds oworder.sh	oworder.exe
oworder2.tm2	Builds oworder2.sh	oworder2.exe
oworder3.tm2	Builds oworder3.sh	oworder3.exe
oworder4.tm2	Builds oworder4.sh	oworder4.exe
oworder5.tm2	Builds oworder5.sh	oworder5.exe
oworder6.tm2	Builds oworder6.sh	oworder6.exe
oworder7.tm2	Builds oworder7.sh	oworder7.exe

For a Windows system, a temporary file is not created. The original file is copied and renamed to the destination file.

Original File	Destination File
OWMENU.TMP	OWMENU.BAT
OWARMENU.TMP	OWARMENU.BAT
OWINMENU.TMP	OWINMENU.BAT
OWSOMENU.TMP	OWSOMENU.BAT
OWLOOKUP.TMP	OWLOOKUP.BAT
OWORD2.TMP	OWORD2.BAT
OWORD6.TMP	OWORD6.BAT
OWINVPR.TMP	OWINVPR.TMP
OWPBRK.TMP	OWPBRK.BAT
OWADDL.TMP	OWADDL.BAT
OWORDER.TMP	OWORDER.BAT
OWORDER2.TMP	OWORDER2.BAT
OWORDER3.TMP	OWORDER3.BAT
OWORDER4.TMP	OWORDER4.BAT

List of Files Copied

OWORDER5.TMP	OWORDER5.BAT
OWORDER6.TMP	OWORDER6.BAT
OWORDER7.TMP	OWORDER7.BAT

For both UNIX/Linux- and Windows-based systems, the CONFIG file is the same and is not renamed.

CONFIG.TMP	config.bbx
-------------------	------------

BB Web Components

Many BB Web components are copied directly to the Remote Access web server. Below is a list of the files copied.:

BBWEB.DOC	GENSMPL1	GENSMPL2	GENSMPL3
BBWEB.CNF	BBWEB.SH	BBWEB2.SH	BBWEBNT.PL
DEMOBBW.BBX	DEMOCENT.BBX	DEMOCLST.BBX	DEMOCLST.TXT
DEMOCUPD.BBX	DEMOILST.BBX	DEMOILST.TXT	DEMOMENU.TXT
DOSCGI.BAT	LICENSE.TXT	README	RSREAD.WBB
RSREADM.WBB	RSTTYS.LCK	RSTTYS.TXT	RSWATCH.SH
RSWATCH.WBB	STDCGI.EXE	UTACTV.WBB	UTADDR.WBB
UTCDATE.WBB	UTCDF.WBB	UTCGL.WBB	UTCGLACC.WBB
UTCGIWIN.WBB	UTCNUM.WBB	UTCPROP.WBB	UTCSTR.WBB
UTENCODE.WBB	UTENTITY.WBB	UTENV.WBB	UTERR.WBB
UTEXIT.WBB	UTFRMGEN.WBB	UTFRMGN2.WBB	UTFRMIN.WBB
UTFRMOUT.WBB	UTHTMFIL.WBB	UTHTMGEN.WBB	UTHTMKEY.WBB
UTHTMOUT.WBB	UTHTMSEL.WBB	UTISIP.WBB	UTMETA.WBB
UTMMOIN.WBB	UTMMOOUT.WBB	UTMORE.WBB	UTSELECT.WBB
UTSEND.WBB	UTSUB.WBB	UTTABLE.WBB	UTTAGS.WBB
UTTEMPFL.WBB	UTPLBAS.WBB	UTPLCP.WBB	UTWATCH.WBB

List of Files Copied

BBWEB.DOC
WINCGI.BAT

GENSMPL1
WINCGI.EXE

GENSMPL2

GENSMPL3

Editing Files

D

You may need to edit certain files to set up remote processing or correct errors. This appendix demonstrates how to edit your CONFIG.BBX file, DOS batch files, and UNIX shell scripts.

Modifying CONFIG.BBX

If you are using multiple drives and Remote Processing you may need to modify CONFIG.BBX to use the **dsksyn** command. If your data lies outside the current directory, you must add the **prefix** command.

CONFIG.BBX

```
stblen=3072
fcbs=70
handles=70
cibs=70
USE_LIM
aliases=20
dsksyn a
dsksyn b
prefix /httpd/ /httpd/bbx/ e:/progrm/
```

There are two important settings to know about when you set up the CONFIG.BBX file: **prefix** and **dsksyn**.

If you have the data files in a different path than the BBx directory, you should specify this in the prefix by entering the full path name of the location where the data files reside. **Make sure that you use forward slashes and end the path with a forward slash ('/')**. If the interpreter is on a different path than the BBx directory, you will need to add that path to the prefix also. Separate multiple prefix entries with a single space.

If a prefix refers to a different drive than the root drive, you must use the **dsksyn** command. This command lets BBx know it can skip looking in a particular drive. For example, imagine you use drives A, B, C, D, and E, and you keep your data files on drive E and server information on drive C. You must enter the **dsksyn** command for drives A, B, and D to avoid errors. When you enter multiple **dsksyn** settings, you must enter each command on a separate line.

Modifying Batch and Script Files

If the interpreter is on a different path than the BBx directory, you will need to change the four .BAT files on DOS; if you are using UNIX, you will need to change the four .EXE scripts.

DOS Changes:

All of the DOS batch files in OSAS Web B2B are similar. The MENU.BAT file is shown here. You can make similar changes to the other batch files.

OWMENU.BAT:

```
@echo off
rem This batch file will work under O'Reilly Website's DOS CGI interface.
rem You can make copies under any name you want, and execute it via
rem http://cgi-dos/name.bat. Each copy can be associated with a BBx
rem program.
rem set BBPROG to be your BBx program, BBEXEC, BBCONFIG, and MEM
rem for rem BBx startup.
set BBPROG=owmenu.bbx
set BBEXEC=c:\server\bbx\vpro5.exe
```

Note

This line must contain the full path to the interpreter.

```
set BBCONFIG=config.bbx
set MEM=1024 -q
set BBTERM=IO
set USERID=osasweb
```

```

if "%os%"=="Windows_NT" goto NT:
rem Win 95 - Website uses OUTPUT_FILE
if "%REQUEST_METHOD%"=="POST" goto WIN95CONTENT
%BBEXEC% -m%MEM% %BBPROG% >%OUTPUT_FILE% - doscgi
goto DONE
:WIN95CONTENT
%BBEXEC% -m%MEM% %BBPROG% <%CONTENT_FILE%
>%OUTPUT_FILE%
goto DONE
:NT
rem Win NT - Website uses stdout
IF "%REQUEST_METHOD%"=="POST" goto WINNTCONTENT
%BBEXEC% -m%MEM% %BBPROG%
goto DONE
:WINNTCONTENT
%BBEXEC% -m%MEM% %BBPROG% <%CONTENT_FILE%
:DONE

```

UNIX changes:

All of the UNIX script files in OSAS Web B2B are similar. The MENU.EXE file is shown here. You can make similar changes to the others script files.

OWMENU.EXE:

```

# Set PGM and MEM to desired values for the bbx program and start size.
# (these can be set in another script, that execs this one.)
PGM=OWCUST.BBX
MEM=512
# modify these values per your bbx configuration
MAXTRIES=5
ERRORLOG=/usr/local/etc/httpd/bbx/error.log
CONFIG=/usr/local/etc/httpd/bbx/config.bbx
BBXEXEC=/usr/local/etc/httpd/bbx/pro5

```

Note

This line must contain the full path to the interpreter.

Editing Files

```
MEM=${MEM:=512}
if [ "$PGM" = "" ]
then
echo "Content-type: text/plain"
echo ""
echo "Invalid pgm argument"
exit
fi
umask 0
RETRIES=0
until [ $RETRIES -gt $MAXTRIES ]
do
$BBXEXEC -q -c$CONFIG -m$MEM $PGM - "$@" 2>/tmp/bberr.$$
if [ -s /tmp/bberr.$$ ]
then
RETRIES='expr $RETRIES + 1'
sleep 1
read msg </tmp/bberr.$$
rm /tmp/bberr.$$ 2>/dev/null
else
RETRIES=99
rm /tmp/bberr.$$ 2>/dev/null
fi
done
if [ ! "$RETRIES" = "99" ]
then
echo `date` $msg >>$ERRORLOG 2>/dev/null
echo "Content-type: text/plain"
echo ""
echo "The database server is too busy. Sorry about that."
echo "Please try again later."
echo ""
echo "By the way, we have recorded this problem, and will add"
echo "capacity if it happens frequently."
fi
```

Files for Remote Access

E

Remote access provides you greater security than the standard OSAS Web B2B setup. By physically separating your company's web server from the data storage, your Internet customers are allowed access to only the information you specify. To do this, you must copy specific files from the main computer—where the OSAS information is stored—to the remote terminal, where the web server resides.

The information stored on the web server is separated into two groups. The first group is the data files; the second group is the public programs.

Copy Data File to Web Server

To copy the data files needed by OSAS Web B2B, use the **Copy Data Files to Web Server** function to copy these files to the remote server:

ARCDx	ARCTx	ARCUx	ARINx
INAIx	INCAx	INCLx	INGLx
INHSx	INLDx	INLOx	INLPx
INLTx	INPPx	INPSx	INQLx
INQTx	INSNx	INTB	INTXx
INUMx	INUPx	INVEx	OSAPPL
OSCN	OSCOMP	OSINFO	OSTPL
OWCC	OWCGx	OWICx	OWIIX
OWGCx	OWTB	RMGCx	RMTDx
RMTHx	SODEx	SOTB	SOTDx
SOTHx			

Files for Remote Access

You also have the option to copy the **ARHix** file, which stores the customer's detailed transaction history. During the initial installation, you must copy this file, which is necessary for the **History** and **Invoice Inquiry** functions. Later, you can skip copying this file, because it can grow quite large and, depending on your computer system, can take quite a long time to copy.

Copy Programs to Web Server

Use the **Copy Programs to Web Server** function on the **Remote Access** menu to copy these programs:.

GENGETOP.PUB	GENOPEN.PUB	GENTAX.PUB
INCALCPR.PUB	INUPD.PUB	INUPDSET.PUB

Basic Web Utility

F

The Basic Web Utility and CGI	F-3
Data Templates	F-17
Sample Program Reference	F-21
Suggestions and Tips	F-37
Standard Structure Options	F-39
HTTP Cookies	F-41
Automatic Session Tracking	F-43
File Uploading Support	F-45
Global String Reference	F-47
Toolkit Program Reference Overview	F-53
Toolkit Program Functional Listing	F-57
rsread.wbb	F-62
rsreadm.wbb	F-63
rswatch.wbb, rswatch.sh	F-64
utaddr.wbb	F-65
utcdate.wbb	F-66
utcdf.wbb	F-67
utcgi.wbb	F-68
utcgiwin.wbb	F-71
utcnum.wbb	F-71
utcprop.wbb	F-71
utcstr.wbb	F-72
utencode.wbb	F-72
utentity.wbb	F-73
utenv.wbb	F-73
uterr.wbb	F-74
utexit.wbb	F-75
utfrmgen.wbb	F-76
utfrmgn2.wbb	F-80
utfrmin.wbb	F-81
utfrmout.wbb	F-82

uthtmfil.wbb	F-85
uthtmgen.wbb	F-87
uthtmkey.wbb	F-90
uthtmout.wbb	F-91
uthtmsel.wbb	F-95
utisip.wbb	F-97
utmeta.wbb	F-98
utmmoin.wbb	F-99
utmmoout.wbb	F-100
utmore.wbb	F-101
utselect.wbb	F-103
utsend.wbb	F-105
utses.wbb	F-106
utsub.wbb	F-106
uttable.wbb	F-107
uttags.wbb	F-108
uttplbas.wbb	F-110
uttempfl.wbb	F-111
uttplcp.wbb	F-111
utwatch.wbb	F-112

The Basic Web Utility and CGI

The popularity of the Internet has led to the development of many software products that take advantage of the capabilities of TCP/IP networking and inexpensive graphical World Wide Web clients called browsers. In addition to use on the Internet, companies have found that these same software products can also be used on their own networks, providing a less expensive alternative to developing graphical applications. These internal networks are called Intranets.

Browsers interpret and display specially formatted text, called HTML, in a graphical format. The document can include links to other documents or files, in-line graphical images, and data input forms that can be used to send information to a World Wide Web server. HTML documents consist of text and markup tags. Examples of tags include **
** (line break), **<p>** (paragraph break), **<h1>** (highest level heading), and **<hr>** (horizontal rule). HTML information is available on the Internet and in many books.

The web server software, or HTTP servers, respond to browser requests for documents and files, and can also perform tasks for browsers through a standard API known as a Common Gateway Interface (CGI). Through CGI, it is possible to produce HTML documents dynamically from database information available on the host system.

The Basic Web Utility provides an easy-to-use interface between the CGI API and Business Basic programs, enabling quick development of web-based applications that read and write legacy data stored in Business Basic data files. There are a large number of uses for this technology, both for internal networks and for businesses that share data with other businesses. For example, a web-based product catalog that comes directly from a live inventory file, with up-to-the-second product availability information, and always-current pricing information.

The Basic Web Utility is a set of CALLable Business Basic programs and platform-specific scripts or programs that interface Business Basic programs using the Basic Web Utility to various web server platforms.

Before You Begin

Before using the Basic Web Utility, a web server must be installed and operating on the same system where Business Basic is operating. The web server must also be configured with a method of invoking CGI scripts (this is the default configuration). An understanding of the naming conventions and directory locations for CGI is essential.

Developers should be familiar with:

- Web browsers, such as Netscape Navigator or Microsoft Internet Explorer.
- HTML and how browsers display HTML documents.
- Business Basic concepts of string templates and global variables or STBL() values.

Basic Web Utility Contents

Toolkit Contents

- UNIX shell scripts for launching Business Basic programs as CGI scripts. Both on-demand and shared directory models are supported.
- Visual PRO/5 version of Business Basic Windows executables for launching either standard (console-based) CGI or WIN-CGI applications.
- Business Basic programs (ut*.wbb) designed to simplify Business Basic CGI programs. Most of these programs are CALLED.
- Business Basic programs (rs*.wbb) to perform simple data transfer over a dedicated serial line, facilitating safe data access to live data from a nearby web server.
- Business Basic programs (rs*.wbb) to perform simple data transfer over a dedicated serial line, facilitating safe data access to live data from a nearby web server.

Installing and Configuring the Basic Web Utility

UNIX

The Basic Web Utility works with PRO/5 on any version of UNIX and with any web server that supports standard CGI. All web servers support standard CGI, including NCSA, Cern, Apache, and Netscape.

To install the Basic Web Utility, copy **bwu.sh** to a CGI directory, or a name recognized as a CGI script (.cgi.). Then, edit the script to start both PRO/5 and the program to run.

Parameter	Description
PGM	The program to run when PRO/5 starts
MEM	An optional parameter defining the amount of memory to start the application with. The default is 512.
MAXTRIES	The number of times the script attempts to start PRO/5 in the event of an error, such as maximum users exceeded.
ERRORLOG	The file to store messages when MAXTRIES is exceeded.
CONFIG	The config.bbx file used by the application.
BBEXEC	The path to the PRO/5 executable.

Windows 95 and Windows NT

The Basic Web Utility for Windows 95 and Windows NT is more complicated to set up and configure than for UNIX. As an overview, a windows executable is run from the web server's scripting directory. This program reads in a configuration file, and executes Visual PRO/5. Visual PRO/5 runs a program that creates the desired web page, and writes the resultant HTML source to a temporary file. The windows executable then passes the contents of this file back to the browser, and the web page appears.

Because the Windows operating system does not handle **stdin** and **stdout** in the same manner as the UNIX operating system, Visual PRO/5 must be run in I/O mode and must utilize temporary files to send information back to the browser. These temporary files are created in the web server's scripting directory, so Visual PRO/5 must have **WRITE** permissions to that directory. Additionally, the web server's home drive must not be **DSKSYN**'d, so that Visual PRO/5 can manipulate these files.

Two separate executables, **stdcgi.exe** and **wincgi.exe**, are provided to manage the process. The **stdcgi.exe** program supports the Standard CGI specification, and is more commonly used. This program should be used when the web server is running Microsoft IIS or Microsoft Personal Web Server for Windows NT or Windows 95. The **wincgi.exe** program is used for web servers that do not support the Standard CGI specification, but rather support the Win CGI specification instead.

When Visual PRO/5 is run on the server, it is executed in the background. This means that **SYSWINDOW** is not be available for debugging purposes. If Visual PRO/5 encounters an untrapped error while running a program, it stops running the program and lists the error in console mode. Since it is in the server's background, this window is not be visible and it appears as though both the Client and web server are hung. In reality, they are simply waiting for the Visual PRO/5 session to complete. Because of this, it is very important that all of the programs trap for unanticipated errors. This is typically done with the **SETERR** verb.

The first step in configuring the utility set is to copy the appropriate cgi executable (**stdcgi.exe** or **wincgi.exe**) to a scripting directory on the Windows web server, typically called **wwwroot/cgi** or **wwwroot/cgi-bin**. The **wwwroot** directory is the root directory of your web server and the **cgi** or **cgi-bin** directory contains scripts and programs that are executed by the web server.

Two associated .bat files, **stdcgi.bat** and **wincgi.bat**, are text files designed to contain configuration parameters used by the executables to specify how Visual PRO/5 should be run. The .bat file extension is used for historical reasons, and these files are never actually executed as a typical batch file. The appropriate .bat file should be copied to the same directory as the .exe file mentioned above. Both the .exe file and .bat file may be renamed to something more descriptive, such as **custquery.exe** and **custquery.bat**. However, they must retain their original extension and the root of the file name must be the same for both files. This allows you to define multiple queries for use with the web server.

The next step is to edit the .bat file to modify the **BBPROG**, **BBEXEC**, **BBCONFIG**, **MEM**, **BBTERM**, and **USERID** parameters. These parameters are used to tell the cgi executable how to find Visual PRO/5 and what startup options to pass to it. Note that when specifying paths for Visual PRO/5, the configuration file, and the program file, all directories with spaces in their names should be changed to reference the short name. For example, if Visual PRO/5 exists as **c:\program files\basis\vpro5\vpro5.exe**, the **BBEXEC** parameter should be set to **c:\progra~1\basis\vpro5\vpro5.exe**.

Parameter	Description
BBPROG	Specifies the program that Visual PRO/5 will run. This program will be responsible for creating the contents of the resultant html page.
BBEXEC	Specifies the full path to the vpro5 executable.
BBCONFIG	Specifies the config.bbx file to be used by Visual PRO/5. The configuration file should have an alias that matches the BBTERM value below. It must also have proper DSKSYN 's and a PREFIX that allows Visual PRO/5 to load the Basic Web Utility programs and the program specified by BBPROG above.
MEM	Specifies the amount of memory with which to start Visual PRO/5.

Parameter	Description
BBTERM	Refers to a terminal alias in BBCONFIG . Often, this has an invisible or minimized mode. Note that some web servers will not allow a console mode window to activate, regardless of this setting. In this case, debugging must be done with error trapping and logging.
USERID	Specifies the userid value to use when connecting to a data server. If no Data Server is required, use a dummy value.

Requisite Knowledge

Before using BB-Web, you must install and run a web server on the same system where Business Basic is operating. The web server needs to be configured with at least one way of invoking CGI scripts (this is always the default configuration anyway, but an understanding of the naming conventions and directory locations for CGI is essential).

You must be familiar with the use of a web browser, such as Netscape Navigator or Microsoft Internet Explorer, and you must have a basic understanding of HTML, and how browsers display HTML documents.

There are scores of good books about HTML as well as many sources of information on the Internet as well.

You must be familiar with Business Basic, and in particular the relatively new concepts of string templates and global variables or **STBL()** values (new, that is, relative to the age of the Business Basic language).

Configuration Requirements and Samples after Installation

To use Basic Web Utility, ensure the following are installed:

- A web server such as Microsoft Personal Web Server (PWS), or IIS (Internet Information Server)

- Visual PRO/5 revision 2.x
- The utility set BASIC WEB UTILITY

Configuration

Visual PRO/5 Configuration

The **config.bbx** file can be modified to contain only minimal information to get Basic Web Utility to work. For example, assuming that Web Server, Basic Web Utility, and Visual PRO/5 reside in different subdirectories on D:\, create a **WebConfig.bbx** file:

```
ALIASES=100
FCBS=128
CIBS=128
STBLEN=4096
SETOPTS 00000060000000
PREFIX D:/BBW/ D:/BBW/progs/ D:/BBW/temp/
alias T0 syswindow
DSKSYN A:
DSKSYN B:
DSKSYN C:
DSKSYN E:
...
DSKSYN Z:
```

In the **stdcgi.bat** file that was copied to **D:\wwwroot\cgi-bin**, make the following changes:

```
set BBPROG=Webtest.bbx
set BBEXEC=d:\basis\vpro5\vpro5.exe
set BBCONFIG=d:\basis\vpro5\WebConfig.bbx
set MEM=1024
set BBTERM=IO
set USERID=dummy
```

If a shortcut is created for testing purposes, set the **Working Directory** to **D:\wwwroot\cgi-bin** because this is where the program resides. Run Visual PRO/5 with the following command line:

```
D:\basis\upro5\upro5.exe -tT0 -m1024 -cD:\basis\upro5\WebConfg.bbx
Enter the following program and save it to D:\BBW\progs\ as Webtest.bbx:
0005 SETERR 50
0010 CALL "utcgi.wbb",ENV$,CGI$,ERRMSG$
0020 CALL "uttags.wbb","h1","My First Web Test Program",HTML$
0030 CALL "utsend.wbb",HTML$
0040 CALL "utexit.wbb",CGI$
0050 RELEASE
```

Executing the CGI Program/Script

Start the browser (Internet Explorer or Netscape), and point it to the web server's **cgi-bin directory/stdcgi.exe**. For example:

```
http://web_server/cgi-bin/stdcgi.exe
```

Once it finds the Web Server and executes the program saved above, the words "My First Web Test Program" appear in a bold header font.

Troubleshooting

Errors can often occur when developing web-based applications, and the above suggestions combined with the following troubleshooting guide can help get an application working quickly.

Error Messages

Remember, all applications should include error trapping to handle any problems that may arise. Errors that can be trapped include user escapes, missing files, and end of file errors. These errors can occur outside the calls to the Basic Web Utility set.

The BASIS Web Utility programs include error trapping as an aid during the development cycle. If any errors occur, a log file (**BWWERR.log**) is created in the current working directory of the program being run. This may not necessarily be the current working directory of PRO/5, but the directory where the script (**stdcgi.exe**) resides.

Errors returned by the **stdcgi.exe** (reported to the browser and the **BWWERR.log**) are:

Error Message	Description
"Can't generate work file 1"	Could not create the first temporary file.
"Can't open work file 1"	Could not open the first temporary file.
"Can't generate work file 2"	Could not create the second temporary file.
"Can't open work file 2"	Could not open the second temporary file.
"Can't open parameter file stdcgi.bat"	Could not open the batch file.
"Couldn't start task!"	Could not start vpro5.(or other exe specified in the batch file).
"Result couldn't be found"	Could not open temp file that contains the output from vpro5.

Errors returned by the **wincgi.exe** (reported to the browser and the **BWWERR.log**) are:

Error Message	Description
"Can't open parameter file wincgi.bat"	Could not open the batch file.
"Sorry: can't start CGI task. Try again later"	Could not start vpro5 (or other exe specified in the batch file).

Debugging Programs during Development

The most common method of debugging a program is to single step through it. (Additional information on the single step verb can be found in the Commands Manual under 'Single Step.')

Always set Byte 1 Bit 08\$ of the options vector before beginning, or errors may occur in public (or called) programs. This also enables console mode and debugging in public programs.

An alternate method of debugging an application is to have the program create dummy files before each step of the program. This method makes it possible to know exactly which utility failed. Using the example from the Configuration section above, a line was added to create a string file before each call to a Basic Web Utility program to help determine exactly which call failed.

```
0005 STRING "d:/bwu/progs/bwustarted",ERR=0050
0010 CALL "utcgi.wbb",ENV$,CGI$,ERRMSG$
0015 STRING "d:/bwu/progs/bwu15",ERR=0050
0020 CALL "uttags.wbb","h1"," My First Web Test Program",HTML$
0025 STRING "d:/bwu/progs/bwu25",ERR=0050
0030 CALL "utsend.wbb",HTML$
0035 STRING "d:/bwu/progs/bwu35",ERR=0050
0040 CALL "utexit.wbb",CGI$
0045 STRING "d:/bwu/progs/bwuworked",ERR=0050
0050 RELEASE
```

CGI - Basic Web Utility Applications

CGI Application

A web server receives a request for a document that is (through naming conventions or directory mapping) an executable program.

The web server establishes an environment and starts the program. In some cases, the program passes data to the standard input handle of the script process. The environment variables, plus the format definition of the data sent on the standard input channel to the executable, are defined by the CGI specification.

The program sends an HTML document back to the server on its standard output handle and then exits. The server, in turn, sends the document back to the browser that made the original request.

Some servers running in non-UNIX environments have additional methods of communicating with the executable. An example is O'Reilly's WebSite for Windows NT and Windows 95, which uses a file-based interface called **Win-CGI**.

Netscape's Fast Start server also supports Win-CGI.

The data passed to the program is normally made up of name=value pairs, passed either through the document request itself or through an HTML form. The program evaluates the names and associated values, performs an action based on the data, and returns an HTML formatted document. For example:

http://www.yourcomp.com/cgi/showcust?custnum=5

In this example, the executable **showcust** is given the data **custnum=5**, which is used to find a customer record and format an HTML document showing customer information.

Because Business Basic interpreters cannot directly communicate with the web server, a middle layer is provided by the Basic Web Utility. There are two methods used by the Basic Web Utility: the on-demand method and the shared directory method.

On-Demand Method

The most efficient method is to start a Business Basic task for each request. Because requests are usually answered quickly, and the task exits, problems with user count restrictions are few. A shell script, called **bwu.sh**, is included with the Basic Web Utility and can be used for this interface method. Also, **bwu.sh** can be copied to a PATHed directory and CGI scripts written:

```
#!/bin/sh
PGM=program.bb; export PGM
MEM=512; export MEM
exec bwu.sh
```

Replace **program.bb** with the Business Basic program name to be started by **bwu.sh**. An alternative is to copy **bwu.sh** to the desired script name in the CGI directory and edit it to include the **PGM** and **MEM** variables internal to that script.

With Windows NT and Windows 95, use **stdcgi.exe** or **wincgi.exe**, depending on the web server's CGI support. Most support standard CGI. Others, such as O'Reilly Website, also support Win-CGI. Associated .bat files must be provided for **stdcgi.exe** and **wincgi.exe** to work.

On-Demand Programming Example

Once the Business Basic program is started, it must read the CGI input and develop a HTML response. A **CALL** to **utcgi.wbb** is needed to read the input. A variety of routines are available to produce the output and should be sent out through **CALLs** to **utsend.wbb**. When the output is complete, a **CALL** to **utexit.wbb** completes the task. Below is an example that builds a list from a small file:

```
call "utcgi.wbb",env$,cgi$,errmsg$
call "uttags.wbb","h1","Data File Listing",html$
dim tpl$:"id:c(5),name:c(30)"
line$="<li> [id] [name]"
chan=unt; open(chan)"data.fil"
call "uthtmfil.wbb", chan, "", "", 0,0,0,"", tpl$,line$,work$, ""
call "uttags.wbb","ul",work$,html$
call "utsend.wbb",html$
call "utexit.wbb",cgi$
release
```

This example was created with the following steps:

1. The CGI input is read.
2. A HTML heading is placed into the variable **html\$**
3. A template is defined for reading data from **data.fil**.
4. A line structure is specified to be used for each record from the file.

5. The file is opened.
6. A list of all records, using the specified **line\$** structure for each, is placed in **work\$**.
7. The "unordered list" tags are added to **work\$**, and the result is added to **html\$**.
8. **html\$**, which now contains a HTML document, is sent back to the user.
9. The task is formatted and released.

Shared Directory Method

A second method uses a shared directory, with a script or other executable started by the web server generating files that contain requests, and one or more Business Basic tasks running in the background, monitoring the same directory and producing results in associated files. This method is less efficient than the on-demand method, but it allows control of user counts, and can also be implemented in environments where Business Basic cannot work with standard input and output handles.

The Business Basic program **utwatch.wbb** can be in the background and watch a directory or run other programs based on requests in that directory. The UNIX shell script **bwu2.sh** can also be used to create shared directory requests.

Shared Directory Programming Example

In the on-demand mode, the Business Basic program is RELEASED, while in the shared directory mode, the program **utwatch.wbb** is repeated until the next request. An option in BBx4/Pro5 installations uses the BACKGROUND directive to start **utwatch.wbb** before completing the request. The program starts more quickly to respond to other requests and takes advantage of the BASIS user count logic of eight background slots per foreground slot. When BACKGROUND is used successfully, the task running in background should RELEASE.

The first attempt to pass control back is with the **background** verb. If successful, the current program is running under a different task, and it sets the **bkg** variable flag to 1. The original task again runs **utwatch.wbb**. Note that **utcgi.wbb** must be called before the background verb because if successful, standard input handles are no longer available. This example is identical to the previous one, except that control is passed back to the program **utwatch.wbb**, rather than to the operating system.

```
call "utcgi.wbb",env$,cgi$,errmsg$
rem "-- try background
bkg=0
background "utwatch.wbb",err=continue
bkg=1
continue:
call "uttags.wbb","h1","Data File Listing",html$
dim tpl:"id:c(5),name:c(30)"
line$="<li> [id] [name]"
chan=unt; open(chan)"data.fil"
call "uthtmfil.wbb", chan, "", "", 0,0,0,"", tpl$,line$,work$, ""
call "uttags.wbb","ul",work$,html$
call "utsend.wbb",html$
call "utexit.wbb",cgi$
rem "-- if background worked, then release
if bkg then release
run "utwatch.wbb"
```

Data Templates

Many of the programs in the Basic Web Utility depend on data definitions found in string templates for data formatting and conversion. HTML, however, is a text format, with no provision to convert data to or from binary formats, or to filter invalid data before it is passed to or received from the server.

The Basic Web Utility automatically performs translations to convert data between HTML text and the internal format used by Business Basic. This is controlled with the data templates, both through the data type included with each template data field, and through user-defined attributes.

The following user-defined attributes may be used to allow the Basic Web Utility to properly handle data stored in various formats:

Text Fields

case=upper
case=lower
case=proper

Force text case on input and output.

caption=phrase

This caption is used by default when generating forms or HTML documents using the **utfrmgen.wbb** and **uthhtmgen.wbb** programs. Captions may not contain spaces because of the way user attributes are accessed. Therefore, the Basic Web Utility converts underscores to spaces ("Customer_Name" becomes "Customer Name").

mask=format

Use the **STR(data:"format")** for output, and **utcstr.wbb** option to map input for text values such as zip codes or phone numbers, where there is formatted external presentation but unformatted internal storage.

passthru=y

If set to **y**, then output data is NOT scanned for correction of HTML entity markup characters, allowing HTML-formatted text to be included in the data.

Text Fields

image=y	If set to y , then output is formatted as a HTML tag.
keys=item-list the formtype,	The uthtmgen.wbb and utfrmgen.wbb programs can format check lists, radio buttons, and selection lists from keys , vals and sep values. These can be placed in the field tags parsed by these programs or can be set to the defaults in the data template. item-list is a list of items keys, with no spaces (use underscores for spaces), delimited by the sepchar value or a semicolon if no sepchar is specified.
vals=desc-list	desc-list is a list of descriptions associated with the keys in item-list . The descriptions cannot contain spaces (use underscores for spaces). Each description is delimited by the sepchar value or a semicolon if no sepchar is specified.
sep=sepchar	sepchar species the delimiter character or characters used in the item-list and desc-list . If the value begins and ends with a \$, it is assumed to be a hexadecimal value.
formtype=form	form-entry-type can be text , memo , radio , checkbox , list , mlist , image , file , or entry type hidden . This serves as the default entry type in form generation. Hidden values are suppressed from text generated by uthtmout.wbb .
wrap=value	For "memo" type fields, adds a wrap=value option to the <textarea> tag.
multisep=sepval	If the data field can store multiple occurrences, sepval defines the separator character used to delimit individual occurrences. If sepval starts and ends with a \$, it is assumed to be a hexadecimal value.

Text Fields**border=pixels**

Specifies that a table used to format checkbox and radio button columns and rows have a border around each cell.

cols=columns

Specifies the default number of columns when generating checkbox, radio buttons, or memo fields in form generation.

rows=rows

Specifies the default number of rows when generating checkbox, radio buttons, selection lists, or memo fields in form generation.

Numeric Fields**scale=n**

Scale number by **n** decimal places. The internal number 10000 becomes 100.00 when **n** is 2. The external number 10.01 becomes 1001 when stored.

mask=format

Use format for output. All numerics not interpreted as dates are processed with **utcnnum.wbb** on input.

Caption=phrase

See Text Fields, above.

Date Fields**date=format**

Dates default to Julian integer format. If format is anything other than a known format, Julian storage is assumed. Custom formats include:

- mdy (MMDDYY)
- mdyy (MMDDYYYY)
- ymd (YYMMDD)
- yymd (YYYYMMDD)
- dmy (DDMMYY)
- dmyy (DDMMYYYY)
- jul (julian integer)
- dtm (julian date with time precision)

Caption=phrase

See Text Fields, above.

Sample Program Reference

Sample Program Reference Overview

The sample programs provided with the Basic Web Utility can be used as learning guides as well as the basis for practical CGI programs.

A UNIX shell script to start a specific Business Basic program, **demobbw.bbx**, establishes the Basic Web Utility CGI environment. It then locates the CGI value of **pgm** to determine which sample program to run. If no **pgm** value is specified, a menu is presented.

The Customer Inquiry application uses the programs **democlst.bbx** and **demoilst.bbx**. These in turn use the HTML document template files **democlst.txt** and **demoilst.txt**.

The Customer Maintenance application uses the programs **democent.bbx** and **democupd.bbx**. The **democent.bbx** links to the customer list program presented in the Customer Inquiry application. These programs do not use HTML templates.

Source code and comments for each of the sample applications are presented on the following pages.

CGI Startup Shell Script

The shell script **bwu.sh** is copied to a **cgi** directory and named **demobbw.sh**. Most servers contain a standard **cgi-bin** directory in the directory mapping structure defined in the server configuration files.

```
#!/bin/sh
# Copyright (C) 1996 by Allen Miglore. All rights reserved.
# bwu.sh is a standard shell script that can be used to start a
# BBx/Pro5 task in a UNIX environment. This script can be copied
# to the cgi directory, given whatever name you need, or it can
# be used as is by other scripts that would set PGM and optionally
```

```

# MEM before exec'ing this script.
# Set PGM and MEM to desired values for the bbx program and start size.
# (these can be set in another script, that execs this one.)
The PGM environment variable must be set to the name of the Business Basic
program (in this case, "demobbw.bbx"). If the
program requires more or less than 512 pages, the MEM variable can also be set.
The values for ERRORLOG , CONFIG , AND BBXEXEC must also be
modified to match the configuration. The file name
ERRORLOG stores messages related to problems starting Business Basic.
CONFIG points to the PRO/5 configuration file used
by Basic Web Utility programs, and BBXEXEC is the name of the PRO/5
executable.
The CONFIG file must be edited so the PREFIX contains the Basic Web Utility
installation directory.
PGM=demobbw.bbx
#MEM=512
# modify these values per your bbx configuration
MAXTRIES=5
ERRORLOG=/usr/lib/pro5/webb/error.log
CONFIG=/usr/lib/pro5/webb/bbweb.cnf
BBXEXEC=/usr/lib/pro5/pro5
MEM=${MEM:=512}
if [ "$PGM" = "" ]
then
echo "Content-type: text/plain"
echo ""
echo "Invalid pgm argument"
exit
fi
umask 0
RETRIES=0
This section starts PRO/5. If it fails, it retries, sends the user an HTML message,
and records the error to the
ERRORLOG file.
until [ $RETRIES -gt $MAXTRIES ]
do
$BBXEXEC -q -c$CONFIG -m$MEM $PGM - "$@" 2>/tmp/bberr.$$
if [ -s /tmp/bberr.$$ ]

```



```

then
RETRIES='expr $RETRIES + 1'
sleep 1
read msg </tmp/bberr.$$
rm /tmp/bberr.$$ 2>/dev/null
else
RETRIES=99
rm /tmp/bberr.$$ 2>/dev/null
fi
done
if [ ! "$RETRIES" = "99" ]
then
echo `date` $msg >>$ERRORLOG 2>/dev/null
echo "Content-type: text/plain"
echo ""
echo "The database server is too busy. Sorry about that."
echo "Please try again later."
echo ""
echo "By the way, we have recorded this problem, and will add"
echo "capacity if it happens frequently."
fi

```

Startup Program - demobbw.bbx

This program is started by the **demobbw.sh** shell script. It loads the CGI environment with a CALL to **utcgi.wbb** and creates the data templates used by the other sample programs.

```

0010 REM "demobbw.bbx - startup for Basic Web Utility demo programs
0020 REM "Copyright 1996 by Allen Miglore. All rights reserved.
0100 REM ^100 - Introduction
0110 REM "This is the startup routine used by the Basic Web Utility demo
0120 REM "program. It reads the CGI input stream, sets up the
0130 REM "data templates, loads a salesperson code table, and then runs
0140 REM "a specified program. The default program is a customer list.
0200 REM ^100 - get cgi inputs
0210 CALL "utaddr.wbb"

```

```

0220 CALL "utcgi.wbb",ENV$,CGI$,ERRMSG$,IF ERRMSG$>"" THEN
GOTO LOADERROR
0300 REM ^100 Dim templates used by the files
0310 DIM
CUST$:"cust_id:c(5*),name:c(30*):case=upper:,addr1:c(30):caption=Address_
1 case=upper:,
addr2:c(30*):caption=Address_2
case=upper:,city:c(20*):case=upper:,state:c(2*):case=upper:,zip:c(5*),
phone:c(10*):mask=(000)-000-
0000:,date_last_sale:c(8):date=yymd:,date_last_pmt:c(8*):date=yymd:,
class:c(2),slsp:c(3):keys={stbl_$SLSPID} vals={stbl_$SLSPNM} sep=\\
formtype=list rows=1:,
terms:c(2*),credit_limit:n(8*):mask=###,##0-
:,ytd_sales:n(10*):mask=###,##0.00-:,
mtd_sales:n(10*):mask=###,##0.00-:,ytd_cost:n(10*):mask=###,##0.00-
:,mtd_cost:n(10*):mask=###,##0.00-:"
0320 DIM
INVOICE$:"cust_id:c(5),invoice:c(5),date:n(8):date=jul:,slsp:c(3),fill:c(2):formt
ype=hidden:,
amount:i(4):scale=2 mask=###,##0.00-:"0330 DIM
SLSP$:"slsp_id:c(3*),name:c(20*)"

```

In the **CUST\$** data template, a list of keys and descriptions are referred to by the global string values **\$SLSPID** and **\$SLSPNM**. These lists are loaded here, and when other routines generate selection lists or lists of salesperson values, the IDs and names are used.

```

0400 REM ^100 - load the STBL's for $SLSPID and $SLSPNM
0410 REM "The CUST$ template references keys and vals for the slsp field.
0420 REM "This facilitates the default use of a select list object for
0430 REM "entry forms, and also will cause HTML merges to use the name
0440 REM "rather than the id for display purposes. Note that placing
0450 REM "these references in the template merely serves to provide
0460 REM "default formatting, and that can be overridden in any given
0470 REM "situation.
0480 LET SLSPCHAN=UNT;OPEN (SLSPCHAN)"GENSMPL2"
0490 LET IDS$="",NAMES$=""
0500 READ RECORD(SLSPCHAN,END=0520)SLSP$

```

```

0510 LET
IDS$=IDS$+SLSP.SLSP_ID$+"\\",NAMES$=NAMES$+SLSP.NAMES$+"\\";
GOTO 0500
0520 LET
X$=STBL("$SLSPID",IDS$),X$=STBL("$SLSPNM",NAMES$),IDS$="",NAMES$=""
0530 CLOSE (SLSPCHAN)

```

Now that the setup is complete, the program branches to the program specified in the CGI input value **pgm**. If no **pgm** value is specified, or an **invalid** **pgm** is specified, the HTML document **demomenu.txt** is loaded and displayed. By using the text “{include filename}” and calling **utmeta.wbb**, a disk-based document can be loaded.

```

0600 REM ^100 - Run the program specified in pgm, or default to the menu.
0610 RUN FIELD(CGI$,"pgm",ERR=0620),ERR=0620
0620 LET TEXT$="{include demomenu.txt}"
0630 CALL "utmeta.wbb", "",TEXT$
0640 CALL "utsend.wbb",TEXT$
0650 DONE: RELEASE
1000 LOADERROR:
REM ^1000 - problem with utcgi - fatal problem
1010 REM "If we are here, there was a REAL BIG PROBLEM. Without
1020 REM "proper initialization, the DOS/WIN environment has no way
1030 REM "of contacting the browser of this, so the best thing is
1040 REM "just to leave gracefully but silently. On UNIX, with
1050 REM "stdout support, a notification of the error can be sent
1060 REM "to the browser.
1070 IF POS("DOS"=INFO(0,0))>0 OR POS("WIN"=INFO(0,0))>0 THEN
GOTO DONE
1080 PRINT "Content-type: text/html", $0A$
1090 PRINT "The following error message was returned by utcgi.wbb:<br>"
1100 PRINT ERRMSG$
1110 GOTO DONE

```

Scrolling Customer Listing - democlst.bbx

This program is used by both sample applications and is the lead program in the Customer Inquiry application in which customers are listed. Each customer has a link to a detailed invoice page. It is also used to select a customer to edit in the Customer Maintenance application.

The program reads up to 10 customer records, in name order, from the customer master file, using key chain 1 (the Name sort key). If additional records are available (there are about 30 in the file), a "More records..." link is provided to continue the list.

```
0001 SETERR ERRTRAP
0010 REM "democlst.bbx - Basic Web Utility Customer Listing Demo
0020 REM "Copyright 1996 by Allen Miglore. All rights reserved.
0030 REM "Permission to derive programs from this is hereby granted,
0040 REM "provided said programs utilize licensed copies of the Basic Web
Utility.
0100 REM ^100 - If restarting this list, the CGI field "startwith"
0110 REM will contain the starting point. This can come
0120 REM either from the form contained at the end of the
0130 REM democlst.txt HTML template, or from the "more records"
0140 REM continuation link.
```

A CGI variable **startwith** is passed to this program in two cases. If the current execution is a continuation of the list, then the "more records" URL will contain this field. In addition, at the bottom of the HTML page is a form with a **startwith** entry field to allow the user to restart the list at any time.

```
0150 LET STARTWITH$="",
STARTWITH$=CVS(FIELD(CGI$,"startwith",ERR=0200),4)
```

These lines set the values for the **uthtmfil.wbb** program, identifying the key chain, records per page, and record structure. The HTML template file, **democlst.txt**, contains a place marker between the **** and **** (unordered list) tags. The structure includes a **** (list element) tag so that each record becomes another bullet item in the document. The name field is treated as a link, and the link is specified in the variable **LINKEXPR\$**, which is constructed to run a program with the CGI value, **cust_id**, set to the customer number of any given record. The program defaults to the sample invoice list **demoilst.bbx** but can be specified in the CGI value **linkto**. The Customer Maintenance application uses the **linkto** value to load and edit a customer record.

```
0200 REM ^100 - Initialize other values
0210 LET KNUM=1,MAXRECS=10
0220 LET STRUCTURE$="<li> [name @link] ([cust_id]), [city], [state] [zip],
[phone]"
0300 REM ^100 - Construct a link expression, used to link each customer
0310 REM to a specified program for further display/edit
0320 LET LINKEXPR$=ENV.SCRIPT_NAME$+"?cust_id=[cust_id
@encode]"
0330 LET
LINKTO$="demoilst.bbx",LINKTO$=FIELD(CGI$,"linkto",ERR=0340)
0340 LET LINKEXPR$=LINKEXPR$+"&pgm="+LINKTO$
0400 REM ^100 - Open Customer Data File
0410 LET CUSTCHAN=UNT;OPEN (CUSTCHAN)"GENSMPL1"
```

After opening the customer master file, the **uthtmfil.wbb** program generates a list of records in the **CUSTLIST\$** variable and lists up to **MAXRECS** records. The list starts with the value in **STARTWITH\$** and continues until the end of the file. If the variable **CURKEY\$** is returned with a value, the end of the file was not reached, and a "More records" URL is constructed to continue where the list ended. This URL is placed in the global string **\$MOREURL**, which is referenced in the HTML text template **democlst.txt**.

```
0500 REM ^100 - Construct List
0510 REM uthtmfil.wbb reads from a channel, and builds list of
0520 REM records, using a particular structure, into a string
0530 CALL "uthtmfil.wbb", CUSTCHAN, STARTWITH$, "", KNUM,
MAXRECS, COUNT, CURKEY$, CUST$,
```

```

STRUCTURE$, CUSTLIST$,LINKEXPR$
0600 REM ^100 - If there are more records in the file, add a "more records"
0610 REM link URL to continue. curkey$ contains the key to the
0620 REM next record required. By placing this in a STBL, the
0630 REM uthtmout.wbb program that loads the HTML template can
0640 REM conditionally place it after the list.
0650 LET X$=STBL("$MOREURL", "");IF CURKEY$="" THEN GOTO 0680
0660 CALL "utencode.wbb",0,CURKEY$,NEXT_STARTWITH$
0670 LET
MOREURL$=STBL("$MOREURL",ENV.SCRIPT_NAME$+"?pgm=democlst.
bbx&startwith="+NEXT_STARTWITH $)

```

The HTML file, **democlst.txt**, is loaded and parsed into the variable **HTML\$**. In that file is a comment "`<!-- List Here -->`", which is substituted with the contents of the **CUSTLIST\$** variable, generated above. The full text of **HTML\$** is sent with the **utsend.wbb** program, and the program exits. In version 1.1, two global string values are created automatically by **uthtmfil**: **\$list** contains the record list returned in **custlist\$**, and **\$moreurl** contains either null (if the end of file is reached), or a URL to continue the list. These can be referenced in **democlst.txt** as **[list]** and **[moreurl]**, respectively.

```

0680 REM ^100 - Load the HTML template, and add the list to it
0690 CALL "uthtmout.wbb",CGI$,"{include democlst.txt}",HTML$,""
0700 CALL "utsub.wbb",HTML$,"<!-- List Here -->",CUSTLIST$
0800 REM ^100 - Send it to the user
0810 CALL "utsend.wbb",HTML$
1000 DONE: REM ^1000 - Clean up and exit
1010 CALL "utexit.wbb",CGI$
1020 IF RETURN_TO_WATCH THEN RUN "utwatch.wbb"
1030 RELEASE

```

It is important to trap all errors because Basic Web Utility programs run in background mode.

```

2000 ERRTRAP: REM ^1000 - generic error trap
2010 CALL "uterr.wbb",ERR,TCB(5),PGM(-2)
2020 GOTO DONE

```

Customer Invoice History - demoilst.bbx

The customer invoice history display is used by the Customer Inquiry application to show a listing of invoices associated with a given customer. The application uses the Scrolling Customer Listing program, and each customer listed is linked to this program through a URL containing the customer ID.

```
0001 SETERR ERRTRAP
0010 REM "demoilst.bbx - Basic Web Utility Customer Invoice Listing Demo
0020 REM "Copyright 1996 by Allen Miglore. All rights reserved.
0030 REM "Permission to derive programs from this is hereby granted, provided
0040 REM "said programs utilize licensed copies of the Basic Web Utility.
```

As with the scrolling customer listing, a **startwith** value can be supplied to begin the invoice listing at a particular point. A "more records" link, containing the **startwith** value, is specified in this program where there are more than 10 invoices to display.

```
0100 REM ^100 - If continuing the invoice list, the CGI field
0110 REM "startwith" is used to indicate what invoice.
0120 LET STARTWITH$="" ,STARTWITH$=FIELD(CGI$,"startwith",
ERR=0130)
```

The invoices are presented in a HTML table structure, so each line from the invoices file is formatted with table row tags. In addition, the table and column heading tags are required above and below the invoice list and are stored in **TABLETOP\$** and **TABLEBOT\$**.

```
0200 REM ^100 - Initialize other values
0210 LET KNUM=0,MAXRECS=10
0215 LET TABLETOP$="<table border><tr><th>Invoice</th><th>Date</th><th>Amount</th></tr>"
0220 LET STRUCTURE$="<tr><td>[invoice]</td> <td>[date]</td> <td align=right>[amount]<br></td></tr>"
0225 LET TABLEBOT$="</table>"
0400 REM ^100 - Open Data Files
0410 LET CUSTCHAN=UNT;OPEN (CUSTCHAN)"GENSMPL1"
0420 LET INVCHAN=UNT;OPEN (INVCHAN)"GENSMPL3"
```

The customer record is read into **CUST\$**, and a table is built from a list of fields in the **CIST\$** template. The HTML text for the table is placed in **CUSTINFO\$**.

```
0450 REM ^100 - Get Customer Data and Create Display
0460 READ RECORD(CUSTCHAN,KEY=FIELD(CGI$, "cust_id",
ERR=NODATA), DOM=0470)CUST$
0470 CALL "uthtmgen.wbb",
```

CUST\$, "cust_id,name,addr1,addr2,city,state,zip,phone,slsp", "", "table border", "", "", **CUSTINFO\$**. A list of invoices (up to **MAXRECS** at a time) is loaded into the variable **INVLIST\$**. The list begins with the customer ID, plus the invoice number indicated in the **STARTWITH\$**, and continues until all invoices for the customer ID have been read. If there are more than **MAXRECS** invoices, the **CURKEY\$** contains that next one in line, after the customer ID is trimmed off. This is used to construct the global string **\$MOREURL**, which is referenced in the HTML text file **demoilst.txt** to allow the user to continue the list. Note the use of the **startwith** value in the URL.

The CGI environment variable **SCRIPT_NAME** contains the name of the cgi script that launched this program. Because the same script is used for all the sample programs, it provides a convenient way of building another URL.

```
0500 REM ^100 - Construct Invoice List
0510 REM uthtmfil.wbb reads from a channel, and builds list of
0520 REM records, using a particular structure, into a string
0530 CALL
"uthtmfil.wbb",INVCHAN,CGI.CUST_ID$+STARTWITH$,CGI.CUST_ID$,K
NUM,
MAXRECS,COUNT,CURKEY$,INVOICES$,STRUCTURE$,INVLIST$,""
0540 IF CURKEY$>"" THEN LET
CURKEY$=CURKEY$(LEN(CGI.CUST_ID$)+1)
0600 REM ^100 - If there are more records in the file, add a "more records"
0610 REM link URL to continue. curkey$ contains the key to the
0620 REM next record required. By placing this in a STBL, the
0630 REM uthtmout.wbb program that loads the HTML template can
0640 REM conditionally place it after the list.
0650 LET X$=STBL("$MOREURL", "");IF CURKEY$="" THEN GOTO 0680
0660 CALL "utencode.wbb",0,CURKEY$,NEXT_STARTWITH$
```



```

0670 LET
MOREURL$=STBL("$MOREURL",ENV.SCRIPT_NAME$+"?pgm=
demoilst.bbx&startwith="+NEXT_STARTWITH$+"&cust_id="+CGI.CUST_I
D$)

```

Both the customer information and the invoice list are substituted into the **demoilst.txt** file, using specific comments as place markers, with the result sent to the user, and the program exits.

```

0680 REM ^100 - Load the HTML template, and add the list to it
0690 CALL "uthtmout.wbb",CGI$,"{include demoilst.txt}",HTML$,""
0700 CALL "utsub.wbb",HTML$,"<!-- Customer Info -->",CUSTINFO$
0710 CALL "utsub.wbb",HTML$,"<!-- Invoice Table -->",
TABLETOP$+$OAS$+INVLIST$+TABLEBOT$
0800 REM ^100 - Send it to the user
0810 CALL "utsend.wbb",HTML$
1000 DONE: REM ^1000 - Clean up and exit
1010 CALL "utexit.wbb",CGI$
1020 IF RETURN_TO_WATCH THEN RUN "utwatch.wbb"
1030 RELEASE
2000 ERRTRAP: REM ^1000 - generic error trap
2010 CALL "uterr.wbb",ERR,TCB(5),PGM(-2)
2020 GOTO DONE

```

Customer Maintenance - democent.bbx

This program generates a customer maintenance form. By default, the form contains no customer data and can be used to create a new customer record. However, if the CGI field **cust_id** is filled in with a valid customer number, that customer record can be edited and updated.

Unlike the prior examples, an HTML document template is not used. Instead, the HTML text is created in the program.

```

0001 SETERR ERRTRAP
0010 REM "democent.bbx - Basic Web Utility demo customer entry form
0020 REM "Copyright 1996 by Allen Miglore. All rights reserved.
0030 REM "Permission to derive programs from this is hereby granted, provided

```

```
0040 REM "said programs utilize licensed copies of the Basic Web Utility.
0100 REM ^100 - Get customer record
```

If the CGI field **cust_id** contains a valid customer number, the customer record is loaded into the data template **CUST\$**. Otherwise, the template is initialized from the startup program.

```
0110 LET CUSTCHAN=UNT;OPEN (CUSTCHAN)"GENSMPL1"
0120 READ RECORD(CUSTCHAN,
KEY=FIELD(CGI$,"cust_id",ERR=0200), ERR=0200)CUST$
```

An entry form is generated from the **CUST\$** data template. The **cust_id** field is hidden and cannot be edited, though it is passed to the **democupd.bbx** program. Fields indicated with a **show** option are display-only. The program **utfrmgn2.wbb** constructs the complete form, including **Submit** and **Reset** buttons, and a form tag to start the **demobbw.sh** script into the variable **CUSTFORM\$**. The non-template field **pgm** is specified as a hidden field. As in URL links, this must be specified for the startup program to run the correct program. The **democupd.bbx** program is used to update the customer file from the contents of the form.

```
0200 REM ^100 - create entry form, with update link to democupd.bbx
0210 LET FLDS$=FLDS$+"cust_id hidden, name, addr1, addr2, city, state,zip,
phone, slsp, date_last_sale show,
date_last_pmt show, credit_limit show, mtd_sales show, ytd_sales show, <input
type=hidden name=pgm
value=democupd.bbx>"
0220 CALL "utfrmgn2.wbb",CUST$,FLDS$, "", "table border", "",
ENV.SCRIPT_NAMES$,"Update record", "Reset
form", 0, CUSTFORM$
```

Multiple calls to **uttags.wbb** build the **HTML\$** document body. Body tags are added, a header is generated, and the complete document is generated into **DOCUMENT\$**. The result is sent to the user, and the program exits. In addition, it is necessary to shuffle **HTML\$** values with a temporary variable to add surrounding tags, such as **<center>** and **</center>** and not double up the text in **HTML\$** because **uttags.wbb** always appends to the last argument.

```

0300 REM ^100 - Use uttags.wbb to generate the rest of the document
0310 CALL "uttags.wbb","h1","Customer Entry Form",HTML$
0320 CALL "uttags.wbb","hr","Enter new customer information, or use the
Customer List link",HTML$
0330 CALL "uttags.wbb","", "to select a customer record to edit.",HTML$
0340 CALL "uttags.wbb","p,href " + ENV.SCRIPT_NAME$ +
"?pgm=democlst.bbx&linkto=democent.bbx",
"Customer List", HTML$
0350 CALL "uttags.wbb","", " - ",HTML$
0360 CALL "uttags.wbb","href "+ENV.SCRIPT_NAME$,"Return to the Menu",
HTML$
0370 REM "msg$ can be sent from democupd.bbx if there is an error
0380 IF MSG$>" THEN CALL "uttags.wbb","h3",MSG$,HTML$
0390 LET WORK$=HTML$,HTML$="";CALL
"uttags.wbb","center",WORK$,HTML$
0400 CALL "uttags.wbb","hr",CUSTFORM$,HTML$
0410 CALL "uttags.wbb","body bgcolor=#E8E8E8",HTML$,BODY$
0420 CALL "uttags.wbb","head,title","Basic Web Utility Demo Customer Entry
Form",HEADING$
0430 CALL "uttags.wbb","html",HEADING$+BODY$,DOCUMENT$
0500 REM ^100 - Send it to the user
0510 CALL "utsend.wbb",DOCUMENT$
1000 DONE: REM ^1000 - Clean up and exit
1010 CALL "utexit.wbb",CGI$
1020 IF RETURN_TO_WATCH THEN RUN "utwatch.wbb"
1030 RELEASE
2000 ERRTRAP: REM ^1000 - generic error trap
2010 CALL "uterr.wbb",ERR,TCB(5),PGM(-2)
2020 GOTO DONE

```

Customer Update - democupd.bbx

This program is launched from the Customer Maintenance form. It reads the CGI input stream for form values and adds or updates the customer record accordingly.

```
0001 SETERR ERRTRAP
0010 REM "democupd.bbx - Basic Web Utility demo customer update from
entry form
0020 REM "Copyright 1996 by Allen Miglore. All rights reserved.
0030 REM "Permission to derive programs from this is hereby granted, provided
0040 REM "said programs utilize licensed copies of the Basic Web Utility.
0100 REM ^100 - Open customer file
0110 LET CUSTCHAN=UNT;OPEN (CUSTCHAN)"GENSMPL1"
```

For a new customer entry, the CGI value **cust_id** contains blanks, and the program jumps to the new customer entry label. An existing customer, selected from the customer list, will have a hidden value on the form that contains the ID, and the current record is read into **CUST\$**.

```
0200 REM ^100 - Get current record
0210 EXTRACT
RECORD(CUSTCHAN,KEY=CGI.CUST_ID$,DOM=NEW_CUSTOMER)CUST$
```

On an existing record, the contents of the form are updated into the contents of the **CUST\$** record using the **utfrmin.wbb** program. The record is then updated, and the program continues.

Concurrency issues are not addressed in the sample programs because the task that loaded the maintenance form cannot retain a lock on the record. There are several methods for managing records that cannot be locked during maintenance. One method is to store a CRC value as a hidden field on the form, calculated from the record used to generate the form. When the current record is loaded in the update program, the CRC values can be compared to determine if a change was made to the record after the form was displayed, and appropriate action can be taken.

```
0300 REM ^100 - Fill in form data
0310 CALL "utfrmin.wbb",CGI$,CUST$
0400 REM ^100 - Update record
0410 WRITE RECORD(CUSTCHAN)CUST$
```

When the customer is updated or added, a message is stored in **MSG\$**, indicating success. The customer entry program is run, which builds a new form for the customer, showing their changes, and also displays the **MSG\$** value so the user can see that the last update was successful.

```
0500 CONTINUE: REM ^100 - redisplay form to show updated data
0510 LET MSG$="Updated "+CVS(CUST.NAME$,3)+"
("+CUST.CUST_ID$+")."
0520 DIM CUST$:FATTR(CUST$)
0530 RUN "democent.bbx"
```

When the CGI value **cust_id** does not contain a valid customer number, a new customer is added. The **CUST\$** data template is filled in from the form CGI data by calling **utfrmin.wbb**. The customer ID is generated by adding 1 to the last key in the file and formatting a new 5-digit key. Once successful, the program continues.

```
0600 NEW_CUSTOMER: REM ^100 - Get new customer ID by adding 1 to
highest current ID
0610 CALL "utfrmin.wbb",CGI$,CUST$
0620 LET LASTKEY$=KEYL(CUSTCHAN,KNUM=0)
0630 LET
CUST.CUST_ID$=STR(NUM(LASTKEY$)+1:FILL(LEN(CUST.CUST_ID$),"0"
))
0640 WRITE RECORD(CUSTCHAN,DOM=NEW_CUSTOMER)CUST$
0650 GOTO CONTINUE
```

Although a normal exit from this program runs **utcent.bbx**, the error routine may need to exit this way.

```
1000 DONE: REM ^1000 - Clean up and exit
1010 CALL "utexit.wbb",CGI$
1020 IF RETURN_TO_WATCH THEN RUN "utwatch.wbb"
1030 RELEASE
2000 ERRTRAP: REM ^1000 - generic error trap
2010 CALL "uterr.wbb",ERR,TCB(5),PGM(-2)
2020 GOTO DONE
```

Suggestions and Tips

There are a variety of ways to write a given program. The following are some basic steps to make CGI programs more portable and easier to maintain.

- Use a single point of entry for CGI applications. Rather than multiple CGI directories with many scripts, a single script to start a single Business Basic program can be used to determine where to go based on a flag passed through the URL or hidden form fields. The benefits to this method include the startup routine performing the initialization, such as calling **utcgi.wbb**, creating templates, and loading code tables.

Parameterize the script name (using **ENV.SCRIPT_NAME\$**) because all routines use the same script. The application becomes more portable to other sites and platforms that may have a different CGI directory structure.

- Use a single point of exit. By using a single exit program, code becomes portable between dynamic and shared-directory methods, as well as other interface methods that may be developed. Ensure that all exits, including error trapping logic, exit through the single routine.
- For tracking large volumes of information between CGI programs, store it in a file rather than a URL and hidden field. Instead of many individual fields, assign and track only a session ID. Keep date and time stamps on session records and periodically clear out old ones. If the application supports cookies, use the automatic session tracking support built into Basic Web Utility 1.1.
- HTML can be generated in programs; however, for complex documents, it is often easier to create HTML files. There are a number of HTML editing programs available that function as word processors. The tags needed by **uthtmout.wbb** and **utfrmout.wbb** can be embedded in the file, and the file referred to by a single **include** phrase in any **uthtm*** or **utfrm*** program:

```
text$="{include filename.txt}"  
call "uthtmout.wbb", tpl$, text$, html$, ""
```

Standard Structure Options

In most web sites it is important to maintain a consistent appearance on all pages, including those generated by CGI. This can include a consistent background color, tiled image, or a standard header and footer on each page.

The Basic Web Utility looks for the the following case-sensitive global strings:

`$title`

A required global string that acts as a trigger. All the other global strings are optional. If the string does not exist, the standard structure is not invoked. If it does exist, the following tags are added on the first call to **utsend.wbb**.

```
<html>
<head>
<title>$title</title>
$otherhead
<body $bodyopt>
$stdhead
```

When **utexit.wbb** is called, and if this global string exists, the following text is added to the document:

```
$stdfoot
</body>
</html>
$othe
rhead
```

A global string to add additional HTML heading elements to the document. A common example would be a **<META ...>** tag, such as **<meta name="keywords" content="sdsi;bbweb">**. To improve viewing of the HTML source, place a line-feed between each header element in the global string.

\$bod
yopt

A global string that, if present, is appended to the **<body>** tag inside the closing bracket. Establishes default colors or background tile images.

\$stdh
ead

A global string to reference a physical disk file that is loaded at the top of each document, just below the **<body>** tag.

\$stdf
oot

A global string references a physical disk file that is loaded at the end of each document by a call to **utexit.wbb**.

To invoke the standard structure, the first call to **utsend.wbb** should be the first body text of the document. It is possible to call **utsend.wbb** with a "Location:" or "Content-type:" header. The Basic Web Utility does not interfere with control of the output or the inclusion of the standard structure elements.

If disk files are used as the page source, and standard structure features are also used, eliminate all heading and closing codes generated by the authoring tool. These codes become redundant and appear in the body of the document.

HTTP Cookies

Netscape Corporation introduced the concept of a "cookie" into their Navigator browser, prompting the specification to be incorporated into other browsers. Cookies provide a mechanism for the server to instruct the browser to retain certain data elements, and to pass those data elements back to the server on demand.

Detailed information about cookies can be found on Netscape Corporation's web site: <http://home.netscape.com>

Cookies are implemented in the Basic Web Utility via global strings. If on the first call to **utsend.wbb**, a global string named **\$cookies** is defined, it is assumed to be a comma-separated list of cookie names. Cookie names refer to other global strings, with names and values passed to the browser through "Set-Cookie:" headers.

For example, if the global string **\$cookies** is set to "fe,fi,fo,fum", the Basic Web Utility looks for four global strings and instructs the browser to store their values. In the Netscape cookie specification, there are additional elements other than data values, such as expiration dates and valid domain and path names. These can be added to the value of the global string if desired, and the Basic Web Utility issues the header "Set-Cookie: name=global-string".

The cookie values are returned automatically by the browser in the CGI input stream, and **utcgi.wbb** looks for these and sets the global string values appropriately.

An example of cookie use would be as a session tracking mechanism. If after calling **utcgi.wbb**, there is no global string **ses_id**, a unique code can be generated and placed into a global string **ses_id**, and that name can be placed in the global string **\$cookies**. From that point, whenever that page is loaded, until the browser is closed on the user's system, the **ses_id** string will be maintained. Using the **expires**, **domain**, and **path** data in the cookie specification, it would be possible for the session ID to be tracked across an entire application for any length of time.

Automatic Session Tracking

The Basic Web Utility 1.1 automatically manages session information using a single HTTP cookie called **\$sesid**. The format of the session information is defined by creating a global string (**STBL**) named **\$sestpl** that contains the string template definition for the session. This string must be defined before the application CALLs **utcgi.wbb** and triggers the internal session management.

```
x$=stbl("$sestpl","name:c(30*),custid:c(10*),items[100]:c(20*")  
call "utcgi.wbb",env$,cgi$,errmsg$
```

In the above example, the session tracks name, custid, and items array fields. Any input from a form or URL that appears in **cgi\$** is automatically updated in the session values.

To retrieve the session record, do one of the following:

```
dim session$:stbl("$sestpl")  
session$=stbl("$sesrec")  
or  
call "utses.wbb",0,session$,errmsg$
```

Use the **utses.wbb** program to store any manipulated session data as follows:

```
session.custid$="1234"  
call "utses.wbb",1,session$,errmsg$
```

Session expiration and deletion is handled automatically based on a 24 hour session lifespan; however, this can be controlled with the global string **\$sesage**, which stores a numeric value of hours. This does two things:

- Records stored on the server are automatically purged when the numeric value is reached. This purge takes place each time **utcgi.wbb** is called with session management triggered. If the value of **\$sesage** is 0 or less, it defaults to 24.

- The cookie used for the session, which is stored on the client (browser) machine, is given an expiration time in hours. If it is set to 0 or less, the browser has no expiration time and expires by default when the user exits the browser.

A session can be manually deleted from the server using the **utses.wbb** program:

```
call "utses.wbb",2,"",errmsg$
```

For session tracking to work, the user account CGI tasks run under must have write rights to the **utses.dat** file. In addition, if the file is not present the first time session tracking is used, the Basic Web Utility creates the file in the Basic Web Utility directory, and, therefore, must have permission to create the file.

File Uploading Support

Newer browsers such as Netscape 3 support a new form posting protocol that allows data files to be uploaded to a web server. For the browser to send the form using a protocol that supports file uploads, the **<form>** tag must have the option "enctype=multipart/form-data" added. When the form is submitted, it is formatted differently. In Basic Web Utility 1.1, **utcgi.wbb** detects this type of form post and loads the **cgi\$** template accordingly. The only difference is that file fields are loaded with a file name, indicating the location of the uploaded file on the server. The name used is a unique generated name that retains the original extension uploaded by the user. The original file name is stored in the **stbl \$filename-x**, where **x** is the name of the field on the HTML form. The files are stored in the system temp directory, or in a directory named in the **stbl \$tempdir**.

See the **utcgi.wbb** reference section for more details.

Global String Reference

Global String (STBL) Usage

\$arraysep	Delimits field names from array element numbers. If \$arraysep is "__", then a cgi name of amount__5 would be interpreted as cgi.amount[5] . The default separator is a hyphen (-), but because this conflicts with Javascript code, it may be necessary to use another value valid in Javascript variable names but one that does not conflict with names used in the application. If the value is changed, it must be set before utcgi.wbb is CALLED.
\$bbweb-version	Stores the version of the Basic Web Utility.
\$bodyopt	Use in standard structure options added to the <body> tag.
\$cgi	Stores the data of cgi\$ for use in programs that do not have access to cgi\$.
\$cgitpl	Stores the string template definition for \$cgi .
\$commdir	Points to the shared directory used by utwatch.wbb and bwu2.sh , if the shared directory model is used by the application. Used by utwatch.wbb and should match the value used by bwu2.sh .
\$cookies	Triggers the placement of cookies on the browser. This must be set to a comma-delimited list of names that reference global string values to be stored as cookies. This must be set before the first CALL to utsend.wbb .
\$scr	The carriage-return portion of a text file line termination. Null on UNIX and ASCII, 13 on Windows.
\$env	Stores the data of env\$ to use in programs without access to env\$.

Global String (STBL) Usage

\$envtpl	Stores the string template definition for \$env .
\$errdesc-errnum	The uterr.wbb program reports the description "An error occurred" for any error value passed to it. It does, however, look for this global string for the description. For example, setting \$errdesc-0 to "A locked file or record was encountered" would report that message for an error 0.
\$filename-fieldname	Stores the user-entered file name when a file upload field is posted from a HTML form. When stored on the web server file system, the Basic Web Utility retains the file extension but renames the file uploaded to a run-time defined name. The original name is stored in this global string.
\$imagelib	Provides a path to the physical location of image files. Any data field with a user attribute of image=y is interpreted by the Basic Web Utility to be an image file. In creating the HTML tag, the Basic Web Utility attempts to open the file. If the file cannot be located by a PREFIX search, define \$imagelib as the physical path to open the file. The Basic Web Utility can then determine the width and height of both .gif and .jpg images and create a tag that is rendered faster by most browsers.
\$imageurl	This is added to the beginning of any tags generated by the Basic Web Utility. If \$imageurl is /home/images/ , and a data field with a user attribute of image=y is 1001.gif , the Basic Web Utility defines the tag as . Note the \$imagelib can also help in determining the width and height for the tag.
\$infile	An internally maintained value used by utwatch.wbb and utcgi.wbb .

Global String (STBL) Usage

\$list	Stores the result of uthtmfil.wbb , uthtmkey.wbb , and uthtmssel.wbb . Use this to merge the results into HTML documents as [\$list] .
\$mailerror	Can be defined on a UNIX system with the mail command to notify an administrator of an error reported by uterr.wbb . It should be set to an email address reachable by the web server. It should not be set on a non-UNIX system.
\$memospacer	If the utmooout.wbb program tries to retain leading spaces, set this global string to y to have the Basic Web Utility add <spacer size=cols> to the text. Browsers that support the <spacer> tag then render the indents.
\$memotable	As with the \$memospacer , this attempts to retain leading spaces on lines, but uses a <table> structure, with <td colspan=cols> tag to control the spacing. Set this global string to y to trigger this behavior.
\$moreurl	Stores a URL suitable for continuation of a list generated by uthtmfil.wbb , uthtmkey.wbb , or uthtmssel.wbb . All values sent in CGI\$ are retained. Suitable for use in HTML text as More records&ldots; .
\$otherhead	Used in standard structure options, added before the <body> tag. Use this tag to add <meta> tags or other tags that go between the <head> and </head> tags (except <title> .)
\$outchan	An internally maintained value used by utsend.wbb .
\$outfile	An internally maintained value used by utwatch.wbb .

Global String (STBL) Usage

\$sesage	If session tracking is on (with the \$sestpl global string), this value determines the age in hours a session cookie lasts. The default is 24. Setting it to 0 results in the session expiring when the browser is closed.
\$sesid	An internally maintained value if session tracking is turned on.
\$sesrec	Stores the current session record, whose structure is defined with \$sestpl .
\$sessioncookie	An internally used value for cookie management.
\$sestpl	Stores the string template definition for session state management. If defined before utcgi.wbb is CALLED, the Basic Web Utility manages a session record using this as the record definition.
\$stdfoot	Used in standard structure options. Points to a file to be appended to the end of all output. Used by utexit.wbb .
\$stdhead	Used in standard structure options. Points to a file to be placed just after the <body> tag is printed by the first call to utsend.wbb .
\$tempdir	Points to a directory to be used for temporary files. If it is not defined, the Basic Web Utility looks for the environment variable TEMP. Then, use the /tmp directory on UNIX, or the current (cgi) directory on Windows. Used by uttempfl.wbb .
\$title	Triggers the standard structure options and is placed in the document header inside <title>&ldots;</title> tags.
\$url	An internally maintained value used in Win-CGI.
\$urltpl	An internally maintained value used in Win-CGI.

Global String (STBL) Usage**\$usechan**

Used to force the Basic Web Utility to search for available channels above this value. If the Basic Web Utility conflicts with other programs used in an application, set this value to a high number to prevent the Basic Web Utility from using the **UNT** value to open files.

\$waitsec

Used by **utwatch.wbb** to pause a specified number of seconds between directory scans. The default is one second.

Toolkit Program Reference Overview

The Basic Web Utility toolkit consists of a number of Business Basic programs and operating system executables. Some of the programs are considered high-level programs and they CALL lower-level programs to perform a task. Some routines may CALL other high-level programs to provide a more complete encapsulation of a task.

Toolkit Program Alphabetical Listing

Program	Description	Syntax
rsread.wbb	Read a record by way of an RS-232 serial connection (UNIX only)	call "rsread.wbb",filename\$,mode\$,ky\$,knum,rec\$,errmsg\$
rsreadm.wbb	Read a record range by way of an RS-232 serial connection (UNIX only)	call rsreadm.wbb",filename\$,key1\$,key2\$,knum,ky\$[all],rec\$[all],count,errmsg\$
rswatch.sh	Watch a RS-232 serial port on a target system (UNIX only)	rswatch.sh -d tty-device -s "stty parameters" -v
utaddr.wbb	ADDR low-level Basic Web Utility Routines	call "utaddr.wbb"
Utcddate.wbb	Convert Text Date to Internal Julian Integer or Date/Time Number	call "utcddate.wbb",strdate\$,juldate
Utcdtf.wbb	Convert Typical and OEM Text Date Formats to/from Julian Date	call "utcdtf.wbb",tojul,format\$,text\$,juldate
Utcgi.wbb	Set Template Variables Based on the Operating System Environment and CGI Input Stream	call "utcgi.wbb",env\$,cgi\$,errmsg\$

Program	Description	Syntax
Utcgiwin.wbb	Win-CGI Support	(Used internally by the Basic Web Utility when the Win-CGI environment is detected and should never be CALLED directly.)
Utcnum.wbb	Convert Formatted Numeric Text String to a Number	call "utcnum.wbb",text\$,value
Utcprow.wbb	Convert a Text String to Proper Case	call "utcprow.wbb",text\$
Utcstr.wbb	Unmask a String, Based on a Mask, and Return an Internal Value	call "utcstr.wbb",text\$,mask\$,result\$
Utencode.wbb	URL Encode and Decode a Text Value	call "utencode.wbb",decode,text\$,result\$
Utentity.wbb	Convert HTML Markup Entities to Browser-Displayable Format	call "utentity.wbb",text\$
Utenv.wbb	Get Environment Variable	call "utenv.wbb",name\$,value\$
Uterr.wbb	Generic Error Display Routine	call "uterr.wbb",errnum,linenum,progrname\$
Utexit.wbb	Clean Up Basic Web Utility Work Files in the Win-CGI Environment	call "utexit.wbb",cgi\$
Utfrmgen.wbb	Generate Form Input Tags from a Template and an Optional List of Fields and Parameters	call "utfrmgen.wbb",tpl\$,flds\$,captions\$,flags\$,captags\$,work\$
Utfrmgn2.wbb	Generate a Complete HTML Form from a Template	call "utfrmgn2.wbb",tpl\$,flds\$,captions\$,flags\$,captags\$,actionval\$,submitval\$,resetval\$,placement,work\$
Utfrmin.wbb	Copy CGI Input to a Data Template	call "utfrmin.wbb",cgi\$,tpl\$

Program	Description	Syntax
Utfrmout.wbb	Merge Template Data Into HTML Form Output	call "utfrmout.wbb",tpl\$,text\$,work\$
Uthtmfil.wbb	Generate HTML List from a File, Given a Key Range and Format Definition	call "uthtmfil.wbb", chan, first\$, last\$, keynum, maxrec, count, curkey\$, tpl\$, text\$, work\$, linkexpr\$
Uthtmgen.wbb	Generate HTML Tags and Text from a Template	call "uthtmgen.wbb",tpl\$, flds\$, captions\$, flags\$, captags\$, linkexpr\$, work\$
Uthtmkey.wbb	Generate HTML List from a File, Given a List of Keys	call "uthtmkey.wbb",chan,keys\$, keylen,tpl\$,text\$,work\$,linkexpr\$
Utfrmout.wbb	Merge HTML Text with Data from a Data Template	call "utfrmout.wbb",tpl\$,text\$,work\$
Uthtmselect.wbb	Generate HTML List from a BBx or PRO/5 Select Verb	call "uthtmselect.wbb",fl\$, where\$, sortby\$, opt\$, maxrec, skip, count, tpl\$,text\$,work\$,linkexpr\$
Utisip.wbb	Verify if the Remote Client Address Is in a List of Valid Addresses	call "utisip.wbb",env\$,iplist\$,valid
Utmeta.wbb	Scans Text For Special Substitutions	call "utmeta.wbb",tpl\$, text\$
Utmoin.wbb	Convert Memo Field (text area tag) CGI Input to Blocked Text	call "utmoin.wbb",cgitext\$, blocked\$,block
Utmooout.wbb	Format Text Containing CR-LF Characters As HTML	call "utmooout.wbb",textin\$,textout\$
Utmore.wbb	Create "More Records" Information, Either As a URL or As Hidden Form Tags	call "utmore.wbb",cgi\$, flds\$, moreurl\$, morehidden\$
Utselect.wbb	Create Select Where Clause from CGI Input Fields	call "utselect.wbb", cgi\$, tpl\$, where\$, andor\$

Program	Description	Syntax
Utsend.wbb	Send Text Back to the Client Browser	call "utsend.wbb",text\$
Utses.wbb	Manage Session Record	call "utses.wbb",mode, sesrec\$,errmsg\$
Utsub.wbb	Substitute Occurrences of Text with Replacement Text	call "utsub.wbb", text\$, from\$, to\$
Uttable.wbb	Create HTML Table from an Array	call "utttable.wbb", caption\$, options\$, dat\$[all], work\$
Uttags.wbb	Append Text with Specified HTML Markup Tags to a HTML String	call "uttags.wbb", tags\$, text\$, html\$
Uttplbas.wbb	Load Data Template from Basis Data Dictionary	call "uttplbas.wbb", dict\$, fl\$, control\$, tpl\$, errmsg\$
Uttempfl.wbb	Create and Open a Temporary String File	call "uttempfl.wbb", tempfile\$, tempchan
Uttplcp.wbb	Copy Matching Values from Source to Destination Templates	call "uttplcp.wbb",source\$, dest\$
Utwatch.wbb	Interface with a Shared Directory CGI Request Process	run "utwatch.wbb"

Toolkit Program Functional Listing

CGI Processing Tools

Program	Description	Syntax
Utcgi.wbb	Set Template Variables Based on the Operating System Environment and CGI Input Stream	call "utcgi.wbb",env\$,cgi\$,errmsg\$
Utsend.wbb	Send Text Back to the Client Browser	call "utsend.wbb",text\$
Uterr.wbb	Generic Error Display Routine	call "uterr.wbb",errnum,linenum,progname\$
Utexit.wbb	Clean Up Basic Web Utility Work Files in the Win-CGI Environment	call "utexit.wbb",cgi\$
Utwatch.wbb	Interface with a Shared Directory CGI Request Process	run "utwatch.wbb"

Data Merging Tools

Program	Description	Syntax
Uthtmout.wbb	Merge HTML Text with Data from a Data Template	call "utfrmout.wbb",tpl\$,text\$,work\$
Utfrmout.wbb	Merge Template Data Into HTML Form Output	call "utfrmout.wbb",tpl\$,text\$,work\$

File Listing Tools

Program	Description	Syntax
Uthtmfil.wbb	Generate HTML List from a File, Given a Key Range and Format Definition	call "uthtmfil.wbb", chan, first\$, last\$, keynum, maxrec, count, curkey\$, tpl\$, text\$, work\$, linkexpr\$
Uthtmkey.wbb	Generate HTML List from a File, Given a List of Keys	call "uthtmkey.wbb", chan, keys\$, keylen, tpl\$, text\$, work\$, linkexpr\$
Uthtmsel.wbb	Generate HTML List from a BBx or PRO/5 Select Verb	call "uthtmsel.wbb", fl\$, where\$, sortby\$, opt\$, maxrec, skip, count, tpl\$, text\$, work\$, linkexpr\$
Utselect.wbb	Create Select Where Clause from CGI Input Fields	call "utselect.wbb", cgi\$, tpl\$, where\$, andor\$

Form and HTML Creation and Capture Tools

Program	Description	Syntax
Utrfmgen.wbb	Generate Form Input Tags from a Template and an Optional List of Fields and Parameters	call "utrfrmgen.wbb", tpl\$, flds\$, captions\$, flags\$, captags\$, work\$
Utrfmgn2.wbb	Generate a Complete HTML Form from a Template	call "utrfmgn2.wbb", tpl\$, flds\$, captions\$, flags\$, captags\$, actionval\$, submitval\$, resetval\$, placement, work\$
Utrfmin.wbb	Copy CGI Input to a Data Template	call "utrfrmin.wbb", cgi\$, tpl\$
Uthtmgen.wbb	Generate HTML Tags and Text from a Template	call "uthtmgen.wbb", tpl\$, flds\$, captions\$, flags\$, captags\$, linkexprs\$, work\$

Other CGI Tools

Program	Description	Syntax
Utencode.wbb	URL Encode and Decode a Text Value	call "utencode.wbb",decode,text\$,result\$
Untity.wbb	Convert HTML Markup Entities to Browser-Displayable Format	call "untity.wbb",text\$
Utmeta.wbb	Scans Text For Special Substitutions	call "utmeta.wbb",tpl\$, text\$
Utmore.wbb	Create "More Records" Information, Either As a URL or As Hidden Form Tags	call "utmore.wbb",cgi\$, flds\$, moreurl\$, morehidden\$
Utses.wbb	Manage Session Record	call "utses.wbb",mode,sesrec\$,errmsg\$
Utsub.wbb	Substitute Occurrences of Text with Replacement Text	call "utsub.wbb", text\$, from\$, to\$
Uttable.wbb	Create HTML Table from an Array	call "utable.wbb", caption\$, options\$, dat\$[all], work\$
Utags.wbb	Append Text with Specified HTML Markup Tags to a HTML String	call "uttags.wbb", tags\$, text\$, html\$

Template Tools

Program	Description	Syntax
Utplbas.wbb	Load Data Template from Basis Data Dictionary	call "utplbas.wbb", dict\$, fl\$, control\$, tpl\$, errmsg\$
Utplcp.wbb	Copy Matching Values from Source to Destination Templates	call "utplcp.wbb",source\$, dest\$

Conversion Tools

Program	Description	Syntax
Utcdate.wbb	Convert Text Date to Internal Julian Integer or Date/Time Number	call "utcdate.wbb",strdate\$,juldate
Utcdf.wbb	Convert Typical and OEM Text Date Formats to/from Julian Date	call "utcdf.wbb",tojul,format\$,text\$,juldate
Utcnum.wbb	Convert Formatted Numeric Text String to a Number	call "utcnum.wbb",text\$,value
Utcprop.wbb	Convert a Text String to Proper Case	call "utcprop.wbb",text\$
Utcstr.wbb	Unmask a String, Based on a Mask, and Return an Internal Value	call "utcstr.wbb",text\$,mask\$,result\$

Memo Field Tools

Program	Description	Syntax
Utmoin.wbb	Convert Memo Field (text area tag) CGI Input to Blocked Text	call "utmoin.wbb",cgitext\$,blocked\$,block
Utmooout.wbb	Format Text Containing CR-LF Characters As HTML	call "utmooout.wbb",textin\$,textout\$

RS-232 Serial Connection Tools

Program	Description	Syntax
Rsgread.wbb	Read a record by way of an RS-232 serial connection (UNIX only)	call "rsgread.wbb",filename\$,mode\$,ky\$,knum,rec\$,errmsg\$
Rsgreadm.wbb	Read a record range by way of an RS-232 serial connection (UNIX only)	call "rsgreadm.wbb",filename\$,key1\$,key2\$,knum,ky\${all},rec\${all},count,errmsg\$
Rsgwatch.wbb	Watch a RS-232 serial port on a target system (UNIX only)	rsgwatch.sh -d tty-device -s "stty parameters" -v

Other Tools

Program	Description	Syntax
Utaddr.wbb	ADDR low-level Basic Web Utility Routines	call "utaddr.wbb"
Utenv.wbb	Get Environment Variable	call "utenv.wbb",name\$,value\$
Utisip.wbb	Verify if the Remote Client Address Is in a List of Valid Addresses	call "utisip.wbb",env\$,iplist\$,valid
Uttempfl.wbb	Create and Open a Temporary String File	call "uttempfl.wbb", tempfile\$, tempchan

rsread.wbb

Read a record by way of an RS-232 serial connection

Syntax

```
CALL "rsread.wbb",filename$,mode$,ky$,knum,rec$,errmsg$
```

This program communicates with a running **rswatch.wbb** task on another system. It scans the file for an available port to issue a request on. Each serial port has a corresponding **rswatch** task running on the target system. A library of available serial ports is maintained in the text file **rsttys.txt**.

By using a dedicated serial connection between a live Business Basic system and a Web server system connected to the Internet, potential security issues of having a TCP/IP connection to the live system can be avoided. While a dedicated serial connection is slower and doesn't support concurrency, it is secure.

filename\$ is the name of the file on the target system to access. It can be a full pathname, or any name accessible to the task running **rswatch.wbb** on the target system.

- **mode\$** is the read mode used to access the desired record:
- **n** retrieves the key after that specified in **ky\$** (**ky\$** need not exist).
- **p** retrieves the key before that specified in **ky\$** (**ky\$** need not exist).
- any other value looks for **ky\$** specifically.

ky\$ is the key to access, or a positioning value for the "n" and "p" modes. This returns the actual key of the record returned.

knum is the key chain number to use in a multi-keyed file. If the keys are not unique, the use of **ky\$** for repeated requests may return the same record. It may be necessary to use the **rsreadm.wbb** program to get a range of duplicate secondary keys.

rec\$ returns the data record, with trailing \$00\$ (hex 00) values removed.

errmsg\$ returns an error message if an error occurs. An "Error 2" detects an end-of-file condition when using modes "n" and "p".

rsreadm.wbb

Read a record range by way of an RS-232 serial connection

Syntax

CALL

```
"rsreadm.wbb",filename$,key1$,key2$,knum,ky$[all],rec$[all],count,errmsg$
```

This program communicates with a running **rswatch.wbb** task on another system and scans the file for an available port to issue a request on. Each serial port has a corresponding **rswatch** task running on the target system. A library of available serial ports is maintained in the text file **rsttys.txt**.

By using a dedicated serial connection between a live Business Basic system and a Web server system connected to the Internet, it is possible to avoid potential security issues of having a TCT/IP connection to the live system. While a dedicated serial connection is slower and doesn't support concurrency, it is secure.

filename\$ is the name of the file on the target system to access. It can be a full pathname, or any name accessible to the task running **rswatch.wbb** on the target system.

key1\$ and **key2\$** specify the key range to read and return. All records from key1+\$00\$ through key2+\$FF\$ are included, so "001" for both would successfully return key values from "001001" through "001ZZZ".

knum is the secondary key chain to use if the target file is a multi-keyed file.

ky\$[all] and **rec\$[all]** are arrays that store returned keys and associated records. Values are stored in **ky\$[1..count]** and **rec\$[1..count]**. If count is 0, the the arrays may not be defined.

count contains the number of records returned in the **ky\$[]** and **rec\$[]** arrays, or 0 if no records were found.

errmsg\$ returns an error message if an error occurred.

rswatch.wbb, rswatch.sh

Watch a RS-232 serial port on a target system

This program communicates with requests for records issued by CALLs to **rsread.wbb** and **rsreadm.wbb**. Each rswatch task communicates through one serial port and uses one Business Basic user slot.

Syntax (UNIX)

```
rswatch.sh -d tty-device -s "stty parameters" -v
```

rswatch.sh may be run in background and started by UNIX at boot time by placing a startup script in the **/etc/rc** directory structure.

The **tty** device must be a disabled (logins are not allowed) **tty** device connected to the web server system. If it does not start with **/dev/**, this prefix is automatically added.

The **stty** parameters must be enclosed in quotes if any white space is included and are used to set the baud rate and other communication parameters needed to match the parameters on the Web server serial ports, as specified in the **rsttys.txt** file in the Basic Web Utility directory. The **raw** device mode for the **stty** command must be included.

The default setting if no **-s** option is specified is "9600 raw cs8 ixon ixoff -ixany".

The **-v** option, if specified, causes logging to take place for each transaction and error processed. This can be redirected to a log file running in background.

Note: The **rswatch.sh** script requires editing to correctly start BBx. Modify the values of lines beginning with **bbexec=**, **bbcfnf=**, and **bbweb=** accordingly.

utaddr.wbb**ADDR low-level Basic Web Utility routines**

Syntax

CALL "utaddr.wbb"

Many Basic Web Utility programs make repeated use of several low-level routines. By making those routines resident in memory (ADDR'd), processing performance is improved. CALL this routine once at the beginning of any session that uses Basic Web Utility programs.

utcddate.wbb

Convert text date to internal julian integer or date/time number.

Syntax

```
CALL "utcddate.wbb",strdate$,juldate
```

This low-level routine is used to convert a text string to a julian date value. The current Business Basic default date format is used to determine whether the format is month-day-year, or day-month-year, and the text string is parsed. In BBx and Pro5, the value of STBL("!DATE") is used for this analysis.

The following steps are performed:

- If **strdate\$** is blank, a julian number of 0 is returned.
- If **strdate\$** is @, today's date and time are returned.

If a time is found at the end of **strdate\$**, it is analyzed and converted to a fraction of a day (noon is .5, 6:00 pm is .75) to be added to the julian integer. The time must be in the format **h:mm** and may be followed by a word starting with **p** to force a "pm" assumption.

If possible, month names are replaced with month numbers.

strdate\$ is tested for simple numeric dates, such as 030196 or 11301995, and converted to a julian value.

Finally, **strdate\$** is scanned for three numerics, separated by any non-numeric delimiter, such as 3/1/96 or 11-30-1995, and converted to a julian number.

- If no year is found, the current year is assumed.
- If the conversion cannot be performed, the juldate value is returned as 0.

utcdf.wbb

Convert typical and OEM text date formats to/from julian date

Syntax

```
CALL "utcdf.wbb",tojul,format$,text$,juldate
```

Many Business Basic applications store dates in a string format rather than a julian number. Common formats include fixed digits, such as YYMMDD, and packed decimal, requiring a special formula to unpack into the month, day, and year elements. This low-level routine is designed to convert various formats to and from julian values.

The legal values for **format\$** match those for date fields in data templates. See the Data Templates section for more information.

tojul is set to **0** to convert from **juldate** to **text\$**, or to **1** to convert from **text\$** to **juldate**.

format\$ can be set to any of the following:

```
mdy MMDDYY  
mdyy MMDDYYYY  
ymd YYMMDD  
yyymd YYYYMMDD  
dmy DDMMYY  
dmyy DDMMYYYY  
aon ADD+ON date format  
soa State of the Art (MAS90) date format
```

text\$ stores the stored text version of the date, and **juldate** stores the julian integer version of the date.

utcgi.wbb

Set template variables based on the operating system environment and CGI input stream

Syntax

```
CALL "utcgi.wbb",env$,cgi$,errmsg$
```

With the exception of **utaddr.wbb**, this is the first Business Basic CGI application program called. Once executed, the **env\$** and **cgi\$** string templates can be used to determine environment and CGI values.

The template **env\$** holds the system environment variables, many of which are used in CGI applications. For example, **env.script_name\$** contains the logical name of the executable script that started the BBx session, and **env.remote_addr\$** contains the IP address of the client browser that sent the request to the host server. In addition to creating **env\$** values for each environment variable, a global string (STBL on PRO/5) is created for each environment variable as well.

The **cgi\$** template holds the values of any input values set by the CGI input stream. These are sent to the CGI application in two ways:

The first is through the URL, where name=value pairs are present following a ? character in the URL. For example, this URL would set four values in the **cgi\$** string template:

```
"http://abc.com/cgi-bin/  
runapp?state=CA&type=A+B&zip=&name=A%26P+Market"
```

The following identifies the **cgi\$** string template:

```
cgi.state$="CA"  
cgi.type$="A B"  
cgi.zip$=""  
cgi.name$="A&P Market"
```

The second method is through an HTML form "post", where input form values are passed to the CGI input stream on the stdin file handle (though the Win-CGI specification passes values through a file).

In both methods, the data sent into the CGI application is URL-encoded, all nonprintable and some special characters are converted to hex notation, and spaces are replaced with **+** characters. **utcgi.wbb** decodes these values before placing them in the **cgi\$** template.

In the Win-CGI environment, large values are stored in temporary disk files. If any **cgi\$** variable value begins with a **\$FF\$**, the balance of the value contains a disk file name to store the value sent by the browser. This also makes it important to CALL **utexit.wbb** before exiting the Business Basic environment to remove any temporary files that may have been created.

Names used in both the **env\$** and **cgi\$** string templates are fixed to be valid variable names, if necessary, by replacing any nonletter, nondigit character with an underscore, and by trimming long names to 32 characters. Sequences of more than one underscore are changed to a single underscore. In addition, any names that do not start with a letter are prefixed with an **X_**.

If any CGI input value is in the format **name-n**, then an array element is created in **cgi\$**. All high-level routines understand arrays, and the format **name-n**, when moving data to and from the CGI environment. For example, if a CGI input stream name is **keywords-1**, the **cgi\$** string template variable would be referenced as **cgi.keywords\$[1]**.

The global string **\$arraysep** may be specified before calling **utcgi.wbb** to override the use of a single dash "-" as the element delimiter. For example, setting **STBL("\$arraysep", "_")** causes the name **keywords__1** to be treated as described above. This is important to scripting languages, such as Javascript, that cannot work with elements named with dashes.

This program stores two global strings for use by another program or application, if necessary: **\$cgi** stores the **cgi\$** string, and **\$cgitpl** stores the template definition. To recreate **cgi\$**:

```
dim cgi$:stbl("$cgitpl")
cgi$=stbl("$cgi")
```

File Uploading

Forms that have been specified as "method=post enctype=multipart/form-data" are handled by **utcgi.wbb**. Any field types of "file" are stored in a temporary file, and the file name is stored in the **cgi\$** field. For example, if a form contains a file field named **datafile**, a text field named **name**, and the user filled in **c:\files\customers.dat** for the datafile field, then **cgi\$** might look like this:

```
cgi.name$="xyz"  
cgi.datafile$="/tmp/bw123999.dat"
```

The extension of the uploaded file name is retained, but the name is changed to guarantee its uniqueness. The original file name is stored in a global string **(STBL) \$filename-fieldname**, so the above example would set **stbl("\$filename-datafile")** to **c:\files\customers.dat**. See the **uttempfl.wbb** information for a description of where temporary files are stored.

utcgiwin.wbb

Win-CGI support

This program is used internally by the Basic Web Utility when the Win-CGI environment is detected and should never be CALLED directly.

utcnum.wbb

Convert formatted numeric text string to a number.

Syntax

```
CALL "utcnum.wbb",text$,value
```

This low-level routine converts a text string to a number, ignoring currency symbols and other nonnumeric characters. In PRO/5, it is also sensitive to the OPTS byte 6 position for the decimal character.

The characters "(" and "-" are both interpreted to indicate that the number is negative.

text\$ contains the text to be converted, and **value** returns the numeric value, or 0 if the conversion fails.

utcprop.wbb

Convert a text string to proper case.

Syntax

```
CALL "utcprop.wbb",text$
```

This routine is used by various high-level Basic Web Utility programs when a user attribute **case=proper** is specified in a string template.

text\$ contains the text to be reformatted and returns the formatted text.

utcstr.wbb

Unmask a string, based on a mask, and return an internal value.

Syntax

```
CALL "utcstr.wbb",text$,mask$,result$
```

This program returns a value suitable for reformatting with the same mask, using the **STR()** function. Business Basic developers often store formatted text values without any formatting, and then display or print formatted versions using the **STR()** function. For example, a US zip code stored as "957625555" could be reformatted as "95762-5555" with the mask "00000-0000". This low-level routine examines a formatted **text\$** string and returns the unformatted **result\$** text string by comparing characters in **text\$** to the format specification in **mask\$**.

The Basic Web Utility high-level programs use this routine to convert formatted CGI input when a template specifies a nonnumeric data type and a user attribute of **mask=mask**.

utencode.wbb

URL encode and decode a text value.

Syntax

```
CALL "utencode.wbb",decode,text$,result$
```

The CGI specification requires that certain characters be specially coded to avoid ambiguous usage. This low-level routine may be used to both encode and decode text based on the CGI requirements.

decode is set to **0** to encode **text\$**, returning the result in **result\$**

decode is set to **1** to decode a previously encoded **text\$**, returning the result in **result\$**.

utentity.wbb

Convert HTML markup entities to browser-displayable format.

Syntax

```
CALL "utentity.wbb",text$
```

When sending text to a client browser, it may be necessary to include characters that the browser would normally interpret as HTML markup characters but should be displayed as they are stored. Examples include `<`, `>`, and `&`. This routine converts occurrences of these characters to the proper entity markup. For example, the `<` characters are converted to `<`.

This routine is called automatically for all data elements by the high-level Basic Web Utility programs, unless the user-defined template attribute **passthru=y** is specified.

utenv.wbb

Get environment variable

Syntax

```
CALL "utenv.wbb",name$,value$
```

This low-level routine can be used to access an operating system environment variable. It can be used whether or not **utcgi.wbb** has been called.

name\$ is a case-insensitive environment variable name.

value\$ returns the value of the environment variable.

uterr.wbb

Generic error display routine

Syntax

```
CALL "uterr.wbb",errnum,linenum,progrname$
```

Handling unexpected errors in CGI applications is different from terminal-based applications because there is no direct channel to the end-user. This routine provides a simple method of notifying the user of a problem. Typically, after CALLING this routine, the program exits Business Basic. To call this in PRO/5 use: CALL

```
"uterr.wbb",err,tcb(5),pgm(-2).
```

The routine generates a simple HTML page, displaying a heading and error specifics. The heading displayed for specific error codes can be controlled by creating global string values called **\$errdesc-n**. For example, in PRO/5, the following creates a valid message for a missing file error:

```
dummy$=stbl("$errdesc-12","A missing file error occurred!")
```

In addition to displaying an HTML page, **uterr.wbb** also attempts to mail an administrator user a message about the error. To do this, the global string value **\$mailerror** should be set to the email address of the administrator. With a host running UNIX, this feature opens a pipe to the UNIX "mail" command and sends a message detailing the error.

utexit.wbb**Clean up Basic Web Utility work files in the Win-CGI environment.**

Syntax

```
CALL "utexit.wbb",cgi$
```

When the Basic Web Utility operates in a Win-CGI environment, it may create work files to store large CGI input values. Before exiting, these work files should be erased. This routine examines the CGI string template produced by **utcgi.wbb** and erases any files. It should be CALLED before exiting the Business Basic environment in any Win-CGI installation.

utfrmgen.wbb

Generate form input tags from a template and an optional list of fields and parameters

Syntax

```
CALL "utfrmgen.wbb",tpl$,flds$,captions$,flags$,captags$,work$
```

This high-level program generates the input tags for an HTML form (or part of a form), based on a template and other information. It program is also capable of creating the form as loose text, preformatted and aligned text, or as a bordered or borderless table.

tpl\$ is a data template that contains both the fields and data to generate the form input fields. A variety of user-attributes can be defined in the template to control how field data is handled. See the Data Templates information for an explanation of these attributes.

flds\$ is a text string that contains a comma-delimited list of field names and optional format options. If **flds\$** is null, then a default list is generated from all the fields in **tpl\$**. **flds\$** can refer to individual array elements (**name-n**) or to all array elements (**name-***), in which case the form is generated with all defined array elements separated by a **
** tag within the column. If **flds\$** begins with "*", it is assumed to contain a list of exceptions, so that all the fields in the template **tpl\$** are used to generate the HTML output. Those found in **flds\$** are referenced with the formatting tags included in **flds\$** (for example, if **flds\$** contained the data ***,type checkbox keys="1;2" vals="Type 1;Type 2"**, then all fields in the template would be generated, and the field **type** would be treated as a checkbox input.) See the **utfrmout.wbb** program documentation for information about formatting options.

Any "field" that begins with "<" is assumed to be fixed HTML that is added to the document generated. If a table format is being generated, the field is allocated a full row, using **<td colspan=2>** when creating that particular row.

If **flds\$** begins with "!", then all fields in the template except those listed after the first comma are included in the generated document.

captions\$ is a string template that contains captions for any field to be included in the form (any in **flds\$**, or if **flds\$** is null, then any field in **tpl\$**.) The data for any field found in **captions\$** is used as the caption for the field. If a field to be used on the form is not present in **captions\$**, and the user attribute "caption" is not present in the template, a default caption is generated from the field name by converting underscores to spaces and formatting the name with proper case. A null field in **captions\$** results in no caption for that field.

flags\$ is a text string containing various formatting controls, each separated by at least one space, in any order. The supported flags include:

table	Formats the captions and input fields in table columns, one row per field.
pre	Use pre-format tags to format the form, allowing perfect column alignment without relying on table structures but using fixed pitch fonts.
<p>	Indicates there should be extra space between each row.
align=left, center, right	Sets horizontal alignment for table captions.
cellalign=left, center, right	Sets the horizontal alignment for table data cells.
valign=top, bottom	Sets vertical alignment to "top" or "bottom" for tables.
border border=n	For table forms, there should be a border around each cell.
hdropt=options cellopt=options	Adds options to the <th> and <td> tags, respectively. No spaces are allowed, but underscores can be used for multiple options. hdropt=bgcolor=red_align=right , for example, generates <th bgcolor=red align=right> .

font=fontopts hdrfont=fontopts	Adds a tag to each data cell and caption cell, respectively. Use underscores for spaces. For Example, size=4_color=black adds and tags to the data.
right	Captions are placed to the right of the input fields.
top	Captions are placed above the input fields. In addition, if pre or table is specified, the data is arranged in columns, and array values in the template are scrolled within the column.

captags\$ is a text string that contains any tag used to format the captions before being placed in HTML form. This string is used in a CALL to **uttags.wbb**. Refer to that program page for more information.

work\$ is a text string that returns the HTML version of the form. This form is incomplete as returned and needs a **Submit** button and the **<form> </form>** HTML tags. It may be necessary to use this feature to build a form from several data templates.

The following code creates a simple form:

```
CALL "utcgi.wbb",env$,cgi$,errmsg$
```

```
rem - generate heading
```

```
CALL "uttags.wbb","h1","Customer Update Form",html$
```

```
rem - create template for new customer
```

```
dim rec$:"id:c(5), name:c(40), street:c(40), city:c(30), state:c(2):case=upper:,  
zip:c(9):mask=00000-0000:"
```

```
rem - if existing customer, the rec$ template could be filled here
```

```
rem - create form fields in table format - note ID is a hidden field and can't be  
changed
```

CALL "utfrmgen.wbb",rec\$,"id hidden,name,street,city,state,zip",","",table border","",formhtml\$

rem - add submit button

CALL "uttags.wbb","submit","",formhtml\$

rem - add form tags to form, append to html\$; script to perform update is "/cgi-bin/custupdt"

CALL "uttags.wbb","form /cgi-bin/custupdt",formhtml\$,html\$

rem - send out form in html\$ to browser

CALL "utsend.wbb",html\$

CALL "utexit.wbb",cgi\$

utfrmgn2.wbb

Generate a complete HTML form from a template

Syntax

```
CALL "utfrmgn2.wbb",tpl$,flds$,captions$,flags$,captags$, actionval$,  
submitval$, resetval$, placement,work$
```

This program uses **utfrmgn.wbb** to create a form and adds the **Submit** and **Reset** buttons and form action tags to create a complete HTML input form.

For the **tpl\$**, **flds\$**, **captions\$**, **flags\$**, **captags\$**, and **work\$** variables, refer to the **utfrmgn.wbb** documentation.

actionval\$ contains the name of the script to execute when the **Submit** button is pressed. All forms contain a tag similar to **<form method="post" action="actionname">**. The text in **actionval\$** is used for **actionname**.

submitval\$ contains the caption used on the submit button(s). If this is null, the browser places its own caption on the button.

resetval\$ contains the caption used on the reset button(s). If this is null, no reset button is created on the form.

placement is used to determine where to place the submit and reset buttons. The following values are valid:

- 1 = Below form
- 2 = Above form
- 3 = Both above and below form

Add 100 to these values for a horizontal rule and to separate the buttons from the form body.

If **utfrmgn2.wbb** recognizes any field types of "type=file", the **<form>** tag is created with the **enctype=multipart/form-data** option to support file uploading. **utcgi.wbb** is compatible with this attribute, but only modern browsers support it (i.e., Netscape Navigator 3.0 or above, Microsoft Internet Explorer 3.0 or above.)

utfrmin.wbb

Copy CGI input to a data template

Syntax

```
CALL "utfrmin.wbb",cgi$,tpl$
```

This is a high-level program to copy data from the CGI input template to a data template. It is sensitive to the user-defined format attributes for defining date, number, and string formats. This is typically used to copy data into a data template for storage in a file from either a form being updated, or from a URL that contains CGI input fields.

cgi\$ is the string template generated by **utcgi.wbb**.

tpl\$ is a data template that can incorporate user-defined format attributes. For more information, see the Data Templates section.

utfirmout.wbb

Merge template data into HTML form output

Syntax

```
CALL "utfirmout.wbb",tpl$,text$,work$
```

This high-level program merges BBx data with HTML text and produces a result in **work\$**. The intent is to produce an HTML form with the default handling of data fields as input fields. It is sensitive to the user-defined format attributes in the data template **tpl\$**, as well as format tags in the **text\$** template. The template attributes are discussed in the Data Templates section.

The HTML text string **text\$** may contain any text, meta instruction, or field tag reference. It is evaluated and copied, with possible substitutions and insertions, into **work\$**.

Meta Instructions

{include filename} - loads the text file file name into **text\$**. This method creates HTML template files and incorporates them into an application. **filename** must be accessible to Business Basic through a physical path or prefix search.

{if datafield opr value} ... {end} - evaluates the expression **datafield opr value**. If it fails, it suppresses any text between the **if** and **end** tags. If the expression is valid, the tags are removed, but the text between them is retained.

datafield can be either a field in the data template or a global string variable name. The operator **opr** can be >, <, =, >=, <=, <>, in, or !in. The "in" and "!in" operators ("in" or "not in") perform position checks on text fields and values.

If **datafield** is numeric in the template, the **value** must be a simple number. If **datafield** is defined with a user attribute of "date=format", the **value** should be a text date, such as "11/30/95". **{if}{end}** tags may be nested.

If **datafield** is not found in the data template and is not a global string variable, the meta instruction is retained in the text.

Note that **datafield** should not be enclosed by brackets, and there must be at least one space between **datafield** and **opr**, and **opr** and **value**.

Field tags

Field tags are placed in **text\$** in the format **[dataname type options]**.

type is optional, and if not specified, defaults to "text". If **dataname** is not found in the data template **tpl\$**, and a global string variable is found by that name, it is used as simple text output rather than input.

type can be any of the following words: text, hidden, password, radio, checkbox, memo, list, mlist, show, image, or encode. These are inserted as form input fields in the following formats: text boxes, hidden fields, password input boxes, radio button group, checkbox group, textarea entry, listbox, multi-selection list box, display-only, image tags, and URL-encoded display-only, respectively. URL-encoded display-only format is useful if the insertion is being placed in the middle of a URL link.

The "show" type provides display-only fields in the form.

Radio, checkbox, and list boxes produce groups of selections based on the "keys", "vals", and "sep" options.

Note this regarding image types: unlike uthtmtout.wbb, this program does not look for the "image=y" user attribute.

Therefore, by default, a form displays the field as a text entry box for maintenance of the data contents. The global string **\$imagelib** may be used to specify the physical path to the image files and is prepended to the data from the template to find the image file and calculate pixel dimensions for ".gif" files. Also, the global string **\$imageurl** is prepended to the image src tag: **<image src=" + STBL("\$imageurl") + data-value + ">**options may include:

```
keys="item1;item2;...;itemn"  
vals="desc1;desc2;...;descn"  
sep="char" or $hex$  
cols=columns
```

```
rows=rows  
alt="alternate image description"  
wrap=value
```

The "sep" option defines the separator character in "keys" and "vals". By default, it is a semicolon but may be set to any value, including nonprintable values, except a quote.

The options are used in different ways by the different field input types.

For radio and checkbox groups, "keys" generates the returned value for each group item, and "vals" is used as a caption for each item. In addition, if columns is less than 2, items are positioned vertically. If columns is between 2 and 98, items are placed in a table of the specified number of columns across. If columns is over 98, items are placed horizontally, and the browser breaks the lines in an unpredictable manner.

For list and mlist groups, keys and vals are combined to form each listbox item in the format of **itemn - descn**. In addition, if rows is specified, then the list box is defined as the specified number of rows high.

For memo fields, columns and rows is used to define the text area dimensions. If unspecified, then 60 columns and 15 rows is assumed. Also, the **wrap=value** adds a wrap option to the textarea tags.

For image tag displays, the **alt=** option can be used to specify the alternate description browsers use if the image cannot be displayed.

uthtmfil.wbb

Generate HTML list from a file, given a key range and format definition

Syntax

CALL "uthtmfil.wbb", chan, first\$, last\$, keynum, maxrec, count, curkey\$, tpl\$, text\$, work\$, linkexpr\$

This high-level program simplifies the process of producing an HTML listing of records from a single file, using traditional Business Basic commands to read the file channel for a specified key range, supporting multi-keyed files through their key chain numbers, and supporting a maximum record count, so that the resulting HTML page size can be controlled.

chan is the channel that the file to be read is opened on and must be opened before CALLing this program.

first\$ and **last\$** are text strings that indicate the key range of records to process. In particular, if the user is to "scroll" through the file, **first\$** should be set to the value of **curkey\$** from the last page scrolled through. The program processes all records from **first\$** forward (whether or not **first\$** exists as a key in the file) until a record is read with a key greater than **last\$ + \$\$\$**. If **first\$** or **last\$** are null, the records are processed from the first through the last records, respectively. **first\$** and **last\$** apply to the key chain specified in **keynum**.

keynum is the key chain number used when processing the file.

maxrec indicates the maximum number of records to include in the HTML output.

count returns the actual number of records included in the HTML output.

curkey\$ is returned as null if the end of the file is reached. Otherwise, it contains the value of the key after the last one included in the HTML output. This variable can be used to indicate a starting key value (**first\$**) for the next page in a "scrolling" type of application. Typically, it must be URL encoded before including it in a "more records" URL to continued scrolling. URL encoding is performed by `utencode.wbb`.

tpl\$ is a data template used to read records from the file indicated by **chan**.

text\$ is a string variable that contains the HTML format for each record, along with embedded data name tags, so that a list format or paragraph breaks are generated between records. For each record, the program **uthtmout.wbb** is CALLED, using this as the format text. See the **uthtmout.wbb** information for more details.

work\$ returns the HTML text for the records. The global string **\$list** also stores this value and can be used in CALLs to **uthtmout.wbb**, which can use text containing **[\$list]** to merge the value of **work\$** automatically.

linkexpr\$ is a text string used by **uthtmout.wbb** to generate any href links from the records to another page. See the **uthtmout.wbb** information for more details.

Automatic pagination support

If the **first\$** argument is null, the **cgi\$** variable **bbweb_s** is checked, and, if present, is used as the starting point for the list. On exit, if more records are available, the program generates a global string (STBL) **\$moreurl** that contains the full URL, with any **cgi\$** fields retained and with **bbweb_s** incremented for the next page. This can be used in a hyperlink for a continuation:

```
{if $moreurl > ""}  
<a href="[$moreurl]">More records&ldots;</a>  
{end}
```

uthtmgen.wbb

Generate HTML tags and text from a template

Syntax

CALL "uthtmgen.wbb",tpl\$, flds\$, captions\$, flags\$, captags\$, linkexprs\$, work\$

This high-level program generates a data page in HTML format, based on a template and other information. It is capable of creating data as loose text, preformatted and aligned text, or as a bordered or borderless table.

tpl\$ is a data template that contains both the fields and data to generate the display fields. Various user attributes can be defined in the template to control how field data is handled. See the Data Templates information for a discussion of these attributes.

flds\$ is a text string that contains a comma-delimited list of field names and optional format options. If **flds\$** is null, a default list is generated from all the fields in **tpl\$**. **flds\$** can refer to individual array elements (**name-n**), or it can refer to all array elements (**name-***), in which case the form is generated with all defined array elements separated by a **
** tag within the column. If **flds\$** begins with "*", it is presumed to contain a list of exceptions, so that all the fields in the template **tpl\$** are used to generate the HTML output. Those found in **flds\$** are referenced with the formatting tags included in **flds\$**. For example, if **flds\$** contained the data "*,giffile @image", then all fields in the template would be generated, and the field "giffile" would be treated as an image reference. If **flds\$** begins with "!", then all fields in the template, except those listed after the first comma, are included in the generated document. See the **uthtmout.wbb** information for formatting options.

captions\$ is a string template that can contain captions for any field to be included in the form. Specifically, any in **flds\$**, or if **flds\$** is null, then any field in **tpl\$**. The data for any field found in **captions\$** is used as the caption for the field. If a field to be used on the form is not present in **captions\$**, and a "caption" user attribute is not present in the template string, a default caption is generated from the field name by converting underscores to spaces and formatting the name with the proper case.

Note that a null field in **captions\$** results in no caption for that field. **flags\$** is a text string containing various formatting controls in any order. The flags supported include:

table	Formats the captions and input fields in table columns, one row per field.
tre	Use pre-format tags to format the form, allowing perfect column alignment without relying on table structures but using fixed pitch fonts.
<p>	Indicates there should be extra space between each row.
align=value cellalign=value	For tables, sets horizontal alignment of the caption or cell to value , typically left or right .
valign=value	For tables, sets vertical alignment to value, such as top or bottom.
font=value hdrfont=value	Add and tags to the text in cells and captions, respectively. Because there can be no spaces in value , use underscores for multiple options. For example: font=size=4_color=blue .
cellopt=value hdropt=value	When the table option is used, add value to the end of the <td> and <th> tags, respectively. No spaces are allowed. For multiple options, these are separated with an underscore (see above.)
border border=n	For table forms, there should be a border around each cell.
right	Captions are placed to the right of the input fields.
top	Captions are placed above the input fields. In addition, if the pre or table is specified, the data is arranged in columns, and array values in the template are scrolled within the column.

The text string **captags\$** contains any tag used to format the captions before placing them in the HTML form. This string is used in a CALL to **uttags.wbb**. Refer to that program page for more information.

linkexpr\$ may contain one or more link definitions to use if any data field contains the **@link** lead-in option. If multiple link expressions are contained in **linkexpr\$**, they must be separated by an ASCII character 1 - \$01\$. An example of a link expression links a record in a list to a full-page display of that particular record. Typically, the full-page display is based on the record's key field. Here is an example, assuming there is a script that will be given the records key through the ID field:

```
linkexpr$="/cgi-bin/showpage?ID=[ID @encode]"
```

Note that **uthtmout.wbb** is CALLED recursively to format the actual URL used in the link, so the same formatting information applies to **text\$** and **linkexpr\$**.

work\$ is a text string that returns the HTML document with captions and data fields.

uthtmkey.wbb

Generate HTML list from a file, given a list of keys

Syntax

```
CALL "uthtmkey.wbb",chan,keys$,keylen,tpl$,text$,work$,linkexpr$
```

This high-level program simplifies the process of producing an HTML listing of records from a single file, using a predefined list of keys that can be iterated with a specified key length. The CALLing program builds the list of primary keys and CALLs this program to create HTML for each record specified in that list.

chan is the channel that the file to be READ is opened on must be opened before CALLing this program.

keys\$ is a string variable that contains the list of keys to process. Each key must be the same length, as indicated in the keylen variable.

keylen indicates the length of each key in keys\$

tpl\$ is a data template to read records from the file indicated by **chan**.

text\$ is a string variable that contains the HTML format for each record, along with embedded data name tags. For each record, the program **uthtmout.wbb** is CALLED, using this as the format text. See the **uthtmout.wbb** information for more details.

work\$ returns the HTML text for the records.

linkexpr\$ is a text string used by **uthtmout.wbb** to generate any href links from the records to another page. See the **uthtmout.wbb** information for more details.

uthtmout.wbb

Merge HTML text with data from a data template

Syntax

CALL "uthtmout.wbb",tpl\$,text\$,work\$,linkexpr\$

This program is a high-level routine CALLED by **uthtmfil.wbb**, **uthtmkey.wbb**, and **uthtmsele.wbb**. It can also be CALLED by custom programs when single-file processing is inadequate. It merges Business Basic data with HTML text and produces results in **work\$**. The intent is to produce a standard HTML page, so no input fields are generated as with an input form. If an input form is desired, **utfrmout.wbb** is used.

The program is sensitive to the user-defined format attributes in the data template **tpl\$**, as well as format tags in the **text\$** template. The template attributes are discussed in the Data Templates section. The data template specified can be filled by the CALLing program using data from a number of sources. In applications where multiple data templates do not share data field names, this program can be CALLED multiple times to produce a final HTML output, with different data merged on each CALL.

tpl\$ is a data template used as the source for data field names. It contains data types and user-defined formatting attributes. See the Data Templates section for more information.

text\$ is an HTML text string that may contain any text, meta instruction, or field tag reference. It is evaluated and copied, with possible substitutions and insertions, into **work\$**.

work\$ returns the HTML result of merging **text\$** with data in **tpl\$**.

linkexpr\$ may contain one or more link definitions to be used if any data field contains the **@link** lead-in option. If multiple link expressions are contained in **linkexpr\$**, they must be separated by an ASCII character 1 - \$01\$. An example of a use for a link expression links a record in a list to a full-page display of that particular record. Typically, the full-page display is based on the record's key field. The following example assumes there is a script that will be given the records key through the ID field:

```
linkexpr$="/cgi-bin/showpage?ID=[@encode ID]"
```

If any link expression contains an ASCII 2 (\$02\$) character, the data before the \$02\$ is used as the "href" portion of the link, and data after the \$02\$ is added to the <a ...> tag after the "href=" element. This is helpful when specifying things such as target frames in a link. For example, if the link expression is **/cgi-bin/showpage?id=1"+\$02\$+"target=A**, the generated tag would be **...**.

Note that **uthtmout.wbb** is CALLED recursively to format the actual URL used in the link, so the same formatting information applies to **text\$** and **linkexpr\$**.

Meta Instructions

{include filename} - loads the text file **filename** into **text\$**. This method creates HTML template files and incorporates them in an application. **filename** must be accessible to Business Basic.

{if datafield opr value} ... {end} - evaluates the expression **datafield opr value**. If it fails, any text between the if and end tags is suppressed. If the expression is valid, the tags are removed, but the text between them is retained. **datafield** can be either a field in the data template or a global string variable name. The operator **opr** can be >, <, =, >=, <=, <>, in, or !in. The "in" and "in" operators perform position checks on text fields and values. If **datafield** is numeric in the template, then **value** must be a simple number. If **datafield** is defined with a user attribute of "date=format", then **value** should be a text date, such as "11/30/95". **{if}{end}** tags may be nested.

If **datafield** is not found in the data template and is not a global string variable, the meta instruction is retained in the text.

Note that **datafield** should not be enclosed by brackets, and there must be at least one space between **datafield** and **opr**, and **opr** and **value**.

Field tags

Field tags are placed in **text\$** in the format [**dataname format-control options**].

dataname is a data template field name, or a global string variable name. If it is not found in either form, then it is retained for subsequent CALLs with other data templates.

format-control is an optional format control word used to indicate special formatting of the merged data. These values can optionally be placed before the dataname for compatibility with earlier releases. Valid values are:

@link or **@linkn**, used to create an "data-value" expression from the dataname value. The format of the url-value is specified in the **linkexpr\$** variable. If multiple link formats are specified in **linkexpr\$**, the format **@linkn** can be used, where **n** specifies the link format number to use with this dataname.

@encode forces the dataname value to be URL encoded.

@image causes an **** tag to be generated from the dataname value. If the global string variable **\$imageurl** is defined, it is used as a prefix to the dataname value. In addition, if the global string variable **\$imagelib** is defined, it is used to locate the physical image file, if it resides on the host system, and determine the image dimensions for the **** tag.

@pad ensures that a right padded, rather than trimmed dataname value is used.

@lpad ensures that a left padded, rather than trimmed dataname value is used.

Padding is only necessary on pre-formatted output (such as between **<pre>** and **</pre>** tags.)

options may include:

```
keys="item1;item2;...;itemn"  
vals="desc1;desc2;...;descn"  
sep="char" or $hex$  
alt="image-altname"  
passthru=y or n
```

The "sep" option is used to define the separator character used in "keys" and "vals". By default, it is a semicolon, but may be set to any value, including nonprintable values, except a quote.

If **keys** and **vals** are specified, then the **dataname** value is compared to the list of item values, and the corresponding desc value is merged, rather than the dataname value itself.

If the **@image** lead-in option is specified, the **image-altname** is used to specify the "alt=name" in the HTML image tag. The default name is "image".

Normally, any dataname value that comes from the data template is modified for HTML markup entities. If **passthru=y** is specified, the **utentity.wbb** routine is not CALLED, allowing HTML text to be stored in the data template. The "passthru" user-defined attribute in the data template is overridden by any value specified in the options here.

uthtmlsel.wbb

Generate HTML list from a BBx or PRO/5 SELECT verb

Syntax

```
CALL "uthtmlsel.wbb",fil$, where$, sortby$, opt$, maxrec, skip, count,  
tpl$,text$,work$,linkexpr$
```

This high-level program simplifies the process of producing an HTML listing of records from a single file, using the BBx/Pro5 SELECT verb. It supports a maximum record count and skip record count to control the resulting HTML record set.

The text string **fil\$** contains the name of the file to issue the SELECT from.

The text string **where\$** contains the optional WHERE clause (without the word "WHERE") for use in the SELECT statement.

The text string **sortby\$** contains the optional SORTBY clause (without the word "SORTBY") for use in the SELECT statement.

The text string **opt\$** is used in the SELECT statement's MODE="opt" clause. A common use of **opt\$** would be to control the optimization performed by BBx or Pro5, such as "opt=nosort".

maxrec indicates the maximum number of records to include in the HTML output.

skip indicates the number of records to skip before including records in the HTML output. Because the SELECT statement does not provide a reliable means of starting a list at a specific point in the file, scrolling applications must know how far the previous page had read through the file. This variable is typically transferred in a "more records" URL. It is automatically incremented by the number of records read or returns 0 if the end of the SELECT is reached.

count returns the actual number of records included in the HTML output.

The data template **tpl\$** reads records from the file indicated by **chan**.

The string variable **text\$** contains the HTML format for each record, along with embedded data name tags. For each record, the program **uthtmout.wbb** is CALLED, using this as the format text. See the **uthtmout.wbb** information for more details.

work\$ returns the HTML text for the records. The global string **\$list** also contains this value.

The text string **linkexpr\$** is used by **uthtmout.wbb** to generate any href links from the records to another page. See the **uthtmout.wbb** information for more details.

Automatic pagination support

If the **skip** argument is null, the **cgi\$** variable **bbweb_s** is checked. If present, it is used as the starting point for the list. On exit, if more records are available, the program generates a global string (STBL) **\$moreurl** that contains the full URL, with any **cgi\$** fields retained, and with **bbweb_s** incremented for the next page. This can be used in a hyperlink for a continuation:

```
{if $moreurl > ""}  
<a href="[$moreurl]">More records&ldots;</a>  
{end}
```

This feature assumes that the value of **where\$**, **sortby\$**, and **opt\$** will be duplicated between executions.

utisip.wbb

Verify if the remote client address is in a list of valid addresses

Syntax

```
CALL "utisip.wbb",env$,iplist$,valid
```

One method of implementing security in an Internet application is to verify that the client request appears to be coming from a valid IP address or domain. This low-level routine checks the validity and includes support for wildcards and multiple addresses.

The string template **env\$** is returned by **utcgi.wbb**.

iplist\$ contains a semi-colon delimited list of valid IP addresses and/or domain names. The list may contain wildcards.

For example, **192.0.0.*** would return as valid any IP address that starts with **192.0.0**. Another example: **localhost; *.acme.com** would allow any **localhost** access, plus any systems in the **acme.com** domain.

valid returns **1** if the client address is in **iplist\$**, or **0** if not.

Not all web servers return a domain name, and those that can are often configured not to. However, most servers provide the client IP address.

utmata.wbb

Scans text for special substitutions

Syntax

```
CALL "utmata.wbb",tpl$, text$
```

This program is CALLED by the **uthtmout.wbb** and **utfrmout.wbb** programs and can also be called directly. The purpose of this program is to allow easy file inclusion in text strings and to provide some conditional control over the content of HTML text without resorting to in-line Business Basic coding.

The data template **tpl\$** is used when evaluating conditional suppression. See the Data Templates section for more information.

The text string **text\$** contains the meta commands and also returns the result after substitutions are performed.

{include filename} - loads the text file **filename** into **text\$**. This method creates HTML template files and incorporates them in an application. **filename** must be accessible to Business Basic.

{if datafield opr value} ... {end} - evaluates the expression **datafield opr value**, and if it fails, suppresses any text between the **if** and **end** tags. If the expression is valid, the tags are removed, but the text between them is retained.

datafield can be either a field in the data template or a global string variable name. The operator **opr** can be **>**, **<**, **=**, **>=**, **<=**, **<>**, **in**, or **!in**. The "in" and "in" operators perform position checks on text fields and values. If **datafield** is numeric in the template, **value** must be a simple number. If **datafield** is defined with a user attribute of "date=format", **value** should be a text date, such as "11/30/95". **{if}{end}** tags may be nested. An example follows:

```
{if ytd_sales > 0} <td>Sales: [ytd_sales]</td> {end}
```

If **datafield** is not found in the data template and is not a global string variable, the meta instruction is retained in the text.

Note that **datafield** should not be enclosed in brackets, and there must be at least one space between **datafield** and **opr**, and **opr** and **value**.

utmmoin.wbb

Convert memo field (textarea tag) CGI input to blocked text

Syntax

```
CALL "utmmoin.wbb",cgitext$,blocked$,block
```

Business Basic applications typically store variable length text fields in multiple records, blocked to a certain size.

Often, the maintenance of these blocks is performed one line at a time within the Business Basic application itself.

This routine simplifies the conversion of free form text entry in an HTML form into a format suitable for storage using blocking techniques.

The string variable **cgitext\$** contains data from a **<textarea>** input tag in a HTML form. This information is sent to the server with CR-LF characters wherever the user pressed the **Enter** key while editing the data.

The string text **blocked\$** returns a version of the text passed in **cgitext\$**, blocked based on the block size specified in **block**.

block indicates the block size to use when constructing **blocked\$**

cgitext\$ is analyzed for paragraph boundaries, and all other line breaks are suppressed. The paragraphs are word-wrapped on block boundaries and appended to the **blocked\$** variable. Paragraph breaks are indicated by blank line in **blocked\$**.

utmooout.wbb

Format text containing CR-LF characters as HTML

Syntax

```
CALL "utmooout.wbb",textin$,textout$
```

Web browsers ignore white space when formatting HTML output. To force data output that contains CR-LF line breaks to retain the breaks, include **<P>** and **
** tags in the HTML output. This low-level routine performs this function by analyzing **textin\$** and substituting the appropriate line- or paragraph-breaks.

The string variable **textin\$** contains the data with CR-LF line breaks.

textout\$ returns a HTML version of the text with **<P>** and **
** tags where appropriate.

This routine is not a mirror of **utmoin.wbb**.

utmore.wbb

Create "more records" information, either as a URL or as hidden form tags

Syntax

CALL "utmore.wbb",cgi\$, flds\$, moreurl\$, morehidden\$

When developing record scrolling applications, it may be necessary to provide a link or button to a subsequent set of records through a "more records" link. Often, this type of link only requires one or two fields to be passed to the subsequent task. This information can be easily created manually in Business Basic code. However, if many fields of information passed to the current task must be passed on to subsequent tasks, this routine can simplify the task of creating the HTML text required.

The string template **cgi\$** is generated by **utcgi.wbb**.

flds\$ is an optional, comma-separated list of field names. If specified, only the field names specified are used to generate **moreurl\$** and **morehidden\$** from **cgi\$**. If **flds\$** begins with "!", then all fields in the template, except those listed after the first comma, are included in the generated document.

moreurl\$ is returned as a text string that contains URL-encoded field=value pairs from the values stored in **cgi\$**. For example, if **cgi.state\$="CA"** and **cgi.slsp\$="100"**, then **moreurl\$** would contain **state=CA&slsp=100**. This can then be appended to the end of a cgi link URL, such as **/cgi-bin/listrecs?state=CA&slsp=100**.

morehidden\$ is returned as a set of hidden field input tags, for inclusion in a HTML form. The above example would produce the following text:

```
<input type="hidden" name="state" value="CA">  
<input type="hidden" name="slsp" value="100">
```

This can then be embedded in an HTML form, along with a **Submit** button, to provide another version of a link to a subsequent list.

This routine is sensitive to the presence of two **cgi\$** variables, **bbweb_count** and **bbweb_skip**. If **bbweb_count** is present, **bbweb_skip** is automatically incremented by the value of **bbweb_count** when added to **moreurl\$** and **morehidden\$**.

utselect.wbb

Create SELECT WHERE clause from cgi input fields

Syntax

CALL "utselect.wbb", cgi\$, tpl\$, where\$, andor\$

This program simplifies the parsing of cgi input from an HTML query form. The query form would contain input fields for certain data fields, and once the user filled in and submitted the form, this program would create the WHERE clause of a SELECT statement based on the user entries. It is sensitive to user-defined formatting attributes in the data template.

The string template **cgi\$** is returned by utcgi.wbb.

The data template **tpl\$** is used in the SELECT statement and can include user-defined formatting attributes, as defined in the Data Template section.

where\$ returns the WHERE clause, suitable for use in a SELECT statement, with the syntax "WHERE CPL(WHERE\$)". The assumed data template name used in **where\$** is "rec": rec.state\$="CA". For example: if "rec" is not the right name, a substitution should be performed before executing the SELECT statement.

andor\$ specifies whether the AND or OR boolean is used between fields in **where\$**. If this is not set to "or", then "and" is assumed.

Any field in **cgi\$** and **tpl\$** that isn't null in **cgi\$**, is evaluated. The field name for the where clause expression fragment is taken from **cgi\$**.

If a **cgi\$** field called "opr_name" is found, the where clause operator is taken from that field. Otherwise, the text of the cgi field itself is analyzed for an operator. Operators can be <, >, =, <=, >=, <>, in, like, and between. If there is no operator, but the data in **cgi\$** contains *, ?, or [characters, and the **tpl\$** definition for the field is character rather than numeric, the "like" operator (wildcard) is used. Otherwise, text fields must start with the characters entered, and numeric fields must be equal to the number entered.

If the operator is "between", there must be exactly two occurrences of the same field name, and the where clause will be formed as field >= first-value and field <= second-value.

If the same field occurs multiple times in **cgi\$** (as can happen with checkboxes, multiple selection lists, or intentionally created duplicate named tags), multiple where clause expressions are created with OR boolean logic (other than the between operator case noted above), distinct from the AND or OR booleans used between unique field names.

utsend.wbb

Send text back to the client browser

Syntax

```
CALL "utsend.wbb",text$
```

This high-level routine sends HTML text to the client. It works on all Basic Web Utility platforms, regardless of the output destination, and automatically adds the Content-type: header as needed to the first output.

On the first CALL to **utsend.wbb**, if **text\$** begins with "Location:", **text\$** is sent with a trailing blank line.

On the first CALL to **utsend.wbb**, if **text\$** begins with "Content-type: ", a header is not added by **utsend.wbb**, and the application is responsible for the header and trailing blank line. This enables the application to send any MIME data type to the browser.

utses.wbb

Manage session record

Syntax

```
CALL "utses.wbb",mode,sesrec$,errmsg$
```

This program provides an interface to read, write, and delete the session record, if present. Session tracking can be initiated in the program by creating a global string (STBL) **\$sestpl** before CALLing **utcgi.wbb**. This string is a string template definition to describe what session data the Basic Web Utility should store. The Basic Web Utility generates an HTTP cookie (**\$sesid**) to retain the session ID code that the data is stored under.

mode can be **0** to read the session record, **1** to write it, or **2** to delete it. If **mode** is **0**, **sesrec\$** returns the session record. If **mode** is **1**, it is used to update the session record. It is ignored if **mode** is **2**.

errmsg\$ returns a description if any error occurred. If this value is not blank on exit, there was a problem in session tracking, and the application should take measures to display and/or work around the error.

This program uses the file **utses.dat**, stored in the Basic Web Utility directory and is created automatically if not found. It has a key size of 22 and a record size of 512.

utsub.wbb

Substitute occurrences of text with replacement text

Syntax

```
CALL "utsub.wbb", text$, from$, to$
```

This low-level routine substitutes all occurrences of **from\$** with **to\$** in **text\$**. It is nonrecursive, so **to\$** may contain **from\$**, or conversely, **from\$** may contain **to\$**.

uttable.wbb

Create HTML table from an array

Syntax

CALL "uttable.wbb", caption\$, options\$, dat\$[all], work\$

HTML tables are time consuming to create manually, in part because the markup tags are required for each cell. This high-level routine reduces the effort, if the data for the table can be stored in a 2-dimensional array.

The string variable **caption\$** contains the table caption, if any.

The string variable **options\$** contains the table options for the **<table>** tag. The most common option is "border", or "border=1". Table options are not consistent among browsers, so care must be taken when using them.

dat\$[all] contains the data to be converted to table cells. This must be a 2-dimensional array with columns as the first dimension and rows as the second. If row 0 exists, it is used for column headings; column 0 is similarly used for row headings. All other data is used as regular cells.

work\$ is returned as the HTML table definition.

uttags.wbb

Appends text with specified HTML markup tags, to a HTML string

Syntax

```
CALL "uttags.wbb", tags$, text$, html$
```

This high-level routine is used to add HTML markup tags to text. When there are no structure conflicts, multiple tags can be added with a single CALL by separating the tags with commas. In all cases, tags are applied to **text\$**, and the result is appended to **html\$**, along with a CR-LF sequence. If there are no tags, **text\$** is appended to **html\$**, along with a CR-LF sequence to aid readability.

tags\$ is a comma-separated list of tags, some of which may contain options. In many cases, a tag causes both a pre- and a post-tag to be added to **text\$** because there are many HTML tags with both opening and closing parts.

The tags supported are:

b	adds and
body options	adds <body options> and </body>
br	appends

center	adds <center> and </center>
Comment	adds <!-- and -->
form action	adds <form action="action" method=post> and </form>
form-up	Adds <form action="action" method=post enctype=multipart/form-data> and </form>
h1..6	adds <hn> and </hn>
head	adds <head> and </head>

Hidden name	replaces text\$ with <code><input type="hidden" name="name" value="text\$"></code>
Hr	prepends a <code><hr></code> tag
Href name	adds <code></code> and <code></code>

If name contains an ASCII 2 (\$02\$) character, the data before the \$02\$ is used as the "href" portion of the link, and data after the \$02\$ is added to the `<a ...>` tag after the "href=" element. This is useful for specifying things such as target frames in a link. For example, if the name is `/cgi-bin/showpage?id=1"+"02+"target=A`, the generated tag would be `...`.

Html	adds <code><html></code> and <code></html></code>
I	adds <code><i></code> and <code></i></code>
Img altname	replaces text\$ with <code></code>
P	prepends a <code><p></code> tag to text\$
Pre	adds <code><pre></code> and <code></pre></code>
Rem	same as comment
Reset	replaces text\$ with <code><input type="reset" name="text\$"></code>
Submit value	replaces text\$ with <code><input type="submit" value="value" name="text\$"></code>
Title	adds <code><title></code> and <code></title></code>

text\$ receives pre- and post- html tags in the order listed in **tags\$**, except where **text\$** is replaced.

html\$ returns its original value, plus the formatted value of **text\$**.

uttplbas.wbb

Load data template from Basis data dictionary

Syntax

```
CALL "uttplbas.wbb", dict$, fl$, control$, tpl$, errmsg$
```

This low-level routine is designed to quickly return a template for a file defined in the BASIS data dictionary, which is a directory that contains the BASIS data files, **LOCAT.1**, **FIELD.1**, and **TYPDEF.1**. This program works directly with the dictionary files. There is no need to initialize the extended utility or data dictionary environments.

dict\$ contains the name of the directory where the dictionary files are stored. If empty, the files must be accessible through normal PREFIX-based directory searching.

fl\$ contains the name of the file of the dictionary to be loaded.

control\$ is used internally to track open file channels used by this program. It should be null when this routine is first called, and passed unchanged to additional calls of this program. To close the channels manually, the channel values are stored in **control\$** in the 2-byte binary increments **dec(control\$(1,2))**, **dec(control\$(3,2))**, and **dec(control\$(5,2))**.

tpl\$ returns the data template for the specified file.

errmsg\$ contains an error message if the template generation failed for any reason.

uttempfl.wbb

Create and open a temporary string file

Syntax

CALL "uttempfl.wbb", tempfile\$, tempchan

This low-level routine is used in the Win-CGI environment where temporary files are sometimes needed. It can also be used when files are uploaded and parsed through **utcgi.wbb**, or by the application developer, if a temporary string file is needed.

tempfile\$ returns the name generated for the file. If the environment variable "TEMP" is defined, it is assumed to point to a temporary file directory. Otherwise, in UNIX the "/tmp" directory is used, and finally, the current directory "." is used. To override this directory selection, create a global string (STBL) called **\$tempdir**, and set its value to the directory to be used.

tempchan returns the channel number used to open the file.

uttplcp.wbb

Copy matching values from source to destination templates

Syntax

CALL "uttplcp.wbb", source\$, dest\$

This low-level routine simplifies the task of copying data from one template to another. It scans the destination template, and any matching field names and array elements in the source template are copied.

utwatch.wbb

Interface with a shared directory CGI request process

Syntax

```
RUN "utwatch.wbb"
```

This program continuously scans a directory for files that contain CGI requests. The request file, which must follow the name convention of **filename.in**, will contain the name of a Business Basic program to RUN, along with the environment and CGI data. The specified program is RUN and must use **utsend.wbb** for all its HTML output. When it is finished, it must CALL **utexit.wbb** and RUN **utwatch.wbb**.

Typically, this program runs from one or more background tasks. The more tasks, the more likely that CGI requests will be responded to quickly.

The global variable **\$commdir** should be set to the name of the shared directory before **utwatch.wbb** is run. If it is not set, the current directory is used.

Normally, **utwatch.wbb** scans the shared directory completely and pauses for one second before scanning again. The number of seconds, from 0 to any value allowed by the WAIT directive, can be specified in the global variable **\$waitsec**.

The format of a request file is:

```
program-name  
env-1=val-1  
env-2=val-2  
...  
env-n=val-n  
<blank line>
```

CGI standard input stream (with line-feed appended)

```
---EOF---
```

The first line is the Business Basic program that **utwatch.wbb** should RUN. The next lines, up to a blank line, contain the environment variables and their values. Following the blank line is the CGI input stream that the CGI script received on its standard input handle, with a line feed appended to force an additional line. The last line is the text string "---EOF---

The UNIX shell script **bwu2.sh** provides the interface mechanism for UNIX CGI requests.

Index

A

access codes
 changing, 1-12
 command, 1-12
 entering, 1-12
Account Information
 web function, 8-5
Address Lookup, 1-22, 1-28
address mapping, 1-22, 1-28
addresses, 1-22, 1-28
Aged Trial Balance
 HTMLscreen, 8-7
 web function, 8-7

B

Browse, 1-22
Build Sales Orders from Remote Files
 function, 5-7
 screen, 5-8
Build Sales Orders Log
 sample, 5-9

C

CGI
 defined, 1-5
Change Fields
 function, 3-23
 sample log, 3-26
 screen, 3-23
codes
 access codes, changing, 1-12
commands
 access code, 1-12
 flags, 1-27
 hot keys, 1-27

 Proceed (OK), 1-4
Common Questions, B-1
Copied Files, C-1
 BBX Programs, C-1
Copy Data Files to Web Server
 function, 6-5
 screen, 6-6
Copy OSAS Programs to Web Server
 function, 6-7
 screen, 6-7
COPYPATH table, 3-21
Create Login Page
 function, 4-3
 screen, 4-3
Customer Groups
 customer screen, 3-18
 function, 3-17
 item group screen, 3-17
Customer Groups List
 function, 7-15
 sample, 7-15
Customer History Lookup
 HTML screen, 8-9
 web function, 8-9
Customer Internet Access Codes
 function, 3-13
 screen, 3-9, 3-14
Customer Invoice Inquiry Lookup
 web function, 8-11
Customer Lookup
 HTMLscreen, 8-5
 web function, 8-5
Customer Order Detail Lookup
 HTML screen, 8-20
Customer Order Lookup
 HTML screen, 8-19

Index

web function, 8-19

D

date fields, 1-21

E

Editing Files, D-1

 modifying batch and script files, D-2

 modifying the config.bbx, D-1

EMAIL table, 3-21

 screen, 3-22

error messages, C-1, D-1

F

F2 Inquiry

 button, 1-22

 flag, 1-27

F6 Maintenance

 flag, 1-27

 icon, 1-22

File Maintenance menu structure, 1-8

flags, in text mode, 1-27

G

graphical mode

 drop-down menus, 1-19

 function screens, 1-18

 graphical main menu, 1-16

 Inquiry button, 1-22

 Maintenance icon, 1-22

 MDI menu, 1-17

 shortcut menu, 1-19

 toolbars, 1-21

H

History Inquiry

 web function, 8-9

HOMEPATH table, 3-21

hot keys, 1-27

HTML, 1-5

I

IMGPATH table, 3-21

Inquiry (F2)

 button, 1-22

 flag, 1-27

Install OPEN Web Server Components

 function, 3-21

Install Web Server Components

 function, 4-5

 screen, 4-6

interfaces with Accounts Receivable, 1-9

Internet Access Codes List

 function, 7-13

 sample, 7-13

Internet Companies List

 function, 7-9

 sample, 7-9

Internet Inventory Item Groups List

 function, 7-11

 sample, 7-11

Inventory Additional Description

 HTML screen, 8-17

Inventory Information Lookup

 Group Codes HTML screen, 8-14

 Items HTML screen, 8-15

 Price Breaks HTML screen, 8-18

Inventory Item Pictures

 function, 3-5

 screen, 3-6

Inventory Item Pictures List

 function, 7-7

 sample, 7-7

Inventory Price Break Lookup

 HTML screen, 8-18

Invoice Detail Lookup

 HTML screen, 8-12

Invoice Inquiry Lookup

 HTML screen, 8-11

 invoice detail, 8-12

 web function, 8-11

Item Group Codes

 function, 3-3

Item Group Codes List

 function, 7-5

 sample, 7-5

item groups, 3-3

L

launching

- OSAS in other operating systems, 1-11
- OSAS in Windows, 1-11

Lists

- printing, 7-3

M

main menu

- graphical, 1-16
- MDI, 1-17
- navigating graphical, 1-16
- navigating MDI, 1-17
- navigating text, 1-23
- text, 1-23
- toolbars, 1-21

Maintenance (F6)

- flag, 1-27
- icon, 1-22

mapping, 1-22, 1-28

master access codes

- setting up, 3-13

Master File Lists menu structure, 1-8

Master Lists

- printing, 7-3

menus

- drop-down menus, 1-19
- graphical main menu, 1-16
- MDI, 1-17
- shortcut, 1-19
- structure in OPEN Web, 1-8
- text main menu, 1-23

messages, A-1, C-1, D-1

- in text mode, 1-28

modes

- graphical, 1-15
- text, 1-23

N

navigating

- graphical function screens, 1-18
- graphical main menu, 1-16
- MDI menu, 1-17
- text function screens, 1-25
- text main menu, 1-23

- to directories and files, 1-22

O

OPEN Web

- description, 1-5
- installation checklist, 2-6
- interfaces, 1-9
- menu structure, 1-8
- Requirements and Applications, 2-5

OPEN Web Login page

- HTML screen, 8-3

Order Entry

- Group Codes HTML screen, 8-22
- Line Item HTML screen, 8-24
- Order Entry Items HTML screen, 8-23
- order numbers, 5-4
- Select Shipping Address, 8-27
- Shipping Address Information, 8-28
- Shopping Cart HTML screen, 8-25
- web function, 8-21

Order Entry for Line Item

- HTML screen, 8-24

Order Inquiry

- HTML screen, 8-19
- Order Detail, 8-20
- web function, 8-19

Order Number

- HTML screen, 8-30

order numbers, 5-4

OSAS

- MDI menu, 1-17
- modes, 1-15
 - graphical, 1-15
 - text, 1-23
- starting in other operating systems, 1-11
- starting in Windows, 1-11

OSAS Web Login Page

- web function, 8-3

P

Proceed (OK) command, 1-4

Purge Log File

- function, 5-11
- screen, 5-11

Index

R

- Remote Access, E-1
- Remote Access functions
 - overview, 6-3
- Remote Access menu structure, 1-8
- remote processing
 - defined, 2-5, 5-3, 6-3
 - files copied, C-1
 - function of files copied in OPEN Web, C-4
- right-click menu *See* shortcut menu, 1-19

S

- sales order numbers, 5-4
- Sales Order Processing
 - checklist, 5-3
- Sales Order Processing menu structure, 1-8
- screens
 - graphical function, 1-18
 - graphical main menu, 1-16
 - navigating graphical, 1-18
 - navigating text function, 1-25
 - OSAS MDI menu, 1-17
 - text function, 1-25
 - text main menu, 1-23
- Security Procedures and Devices
 - Firewalls, 2-4
- Setup Item Group Codes
 - screen, 3-3, 5-5
- Setup OPEN Web Components menu structure, 1-8
- shortcut menu, 1-19
- starting
 - OSAS
 - in other operating systems, 1-11
 - in Windows, 1-11
- System Messages, A-1

T

- Tables
 - screen, 3-22
- tables
 - COPYPATH, 3-21
 - EMAIL, 3-21
 - HOMEPATH, 3-21
 - IMGPATH, 3-21
 - WEBPATH, 3-21

Tables List

- function, 7-17
 - sample, 7-17
- text mode
- command line, 1-28
 - commands, 1-27
 - flags, 1-27
 - function screens, 1-25
 - main menu, 1-23
 - messages, 1-28
- toolbars, 1-21
- function screens, 1-21
 - main menu, 1-21
- Transaction Journal
- function, 5-5
 - sample, 5-6

V

- View/Submit Current Order
 - HTML screen, 8-25
 - Order Number HTML screen, 8-30
 - Select Shipping Address HTML screen, 8-27
 - Shipping Address Information HTML screen, 8-28

W

- web pages
 - Account Information, 8-5
 - Aged Trial Balance, 8-7
 - Customer History Lookup, 8-9
 - Customer Invoice Inquiry, 8-11
 - Customer Lookup, 8-5
 - Customer Order Lookup, 8-19
 - History Inquiry, 8-9
 - Invoice Inquiry Lookup, 8-11
 - Order Entry group codes, 8-22
 - Order Entry Items, 8-23
 - Order Entry Line Item, 8-24
 - Order Inquiry, 8-19
 - Order Number, 8-30
 - OSAS Web Login Page, 8-3
 - Select Shipping Address, 8-27
 - Shipping Address Information, 8-28
 - View Shopping Cart, 8-25
- WEBPATH table, 3-21
- workstation date, 1-13