Patch 6.0-TSD-0007

Tivoli Problem Management

End-User Web Interface Administrator's Guide

Tivoli Problem Management

End-User Web Interface Administrator's Guide

Tivoli Problem Management End-User Web Interface Administrator's Guide (Feb, 2000)

Trademarks

The following product names are trademarks of Tivoli Systems or IBM Corporation: IBM, Intelligent Miner for Text, OS/2, DB2/2, Tivoli Asset Management, Tivoli Change Management, Tivoli Decision Support, Tivoli Problem Management , Tivoli Service Desk, and Tivoli Service Desk Developer's Toolkit. Microsoft, Windows, Windows NT, Windows 95, Windows 98, and the Windows logo are trademarks or registered trademarks of Microsoft Corporation. UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited. Crystal Reports is a trademark of Seagate Software, Inc. Other company, product, and service names mentioned in this document may be trademarks or servicemarks of others.

Notices

References in this publication to Tivoli Systems or IBM products, programs, or services do not imply that they will be available in all countries in which Tivoli Systems or IBM operates. Any reference to these products, programs, or services is not intended to imply that only Tivoli Systems or IBM products, programs, or services can be used. Subject to Tivoli Systems' or IBM's valid intellectual property or other legally protectable right, any functionally equivalent product, program, or service can be used instead of the referenced product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by Tivoli Systems or IBM, are the responsibility of the user.
Tivoli Systems or IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594. Order Number:

Contents

Preface

This book covers the installation and maintenance tasks that are necessary to successfully implement Tivoli Problem Management End-User Web Interface. As an administrator of a system or network, you should be familiar or have experience with the following:

- The Internet and your company's intranet
- Internet communication protocols
- Your network architecture
- TPM functions and terms

About this Book

The *Tivoli Problem Management End-User Web Interface Administrator's Guide* provides information in the following areas:

- Installation of administration application on the Application Server
- Operation of the TPM End-User Web Administration Notebook
- Configuration of the client and administration application of TPM End-User Web Interface

Note: Error messages for this application can be found on the Tivoli Support Center web site. (For more information, see the section "Contacting Customer Support" on page xiii.)

What you will be able to do

By reading this book, you will be able to do the following:

- Define separate capabilities for your employees (intranet users) compared to customers of your company's products (internet users)
- Maintain user capabilities, privileges, and diagnostics
- Customize the client application to fit the needs of your customers

Topics in this book

The *Tivoli Problem Management End-User Web Interface Administrator's Guide* contains the following chapters and appendices:

- Chapter 1, "Introduction to TPM End-User Web Interface," introduces the system architecture and components, and describes a typical TPM End-User Web Interface client session.
- Chapter 2, "Installing TPM End-User Web Interface," provides prerequisites and instructions to install the administration application of TPM End-User Web Interface, as well as the necessary web server and web daemon files.
- Chapter 3, "Configuring TPM End-User Web Interface," describes the configuration of request, security, and account settings.
- Chapter 4, "Configuring the Web Communication Layer," presents a sample web communication layer file (webcomm.ini) as well as argument definitions in the .ini file.
- Chapter 5, "Configuring the CGI Scripts," presents a sample TPM End-User Web Interface perl script (web_access.cfg) as well as argument definitions in the .cfg file.
- Chapter 6, "Troubleshooting," describes solutions to common problems that may occur when implementing TPM End-User Web Interface.
- Appendix A, "TPM Tables Used by End-User Web Interface," is a reference to the TPM tables with which TPM End-User Web Interface interacts.
- Appendix B, "Command Codes," contains reference tables for command and subcommand codes for the web communication layer, CGI perl scripts, and Application Server.
- Appendix C, "Configuring the End-User Web Administration Notebook," describes how to add certain fields and hypermedia objects to the End-User Web Administration Notebook.

Other Books in the TSD Library

In addition to this book, other books and documentation in the Tivoli Service Desk 6.0 library include the following:

- *Tivoli Asset Management Utilities Guide*

  This book explains the different utilities that system integrators or database administrators can use to manage the asset database, and integrate data from supported network management applications.

- *Tivoli Change Management User's Guide*

  This book describes the basic features and processes of TCM, including creating and approving changes and completing other tasks related to changes.

- *Tivoli Problem Management User's Guide*

  This book explains the TPM features and procedures used by help desk analysts, including registering calls and resolving problems.

- *Tivoli Service Desk Administrator's Guide*

  This book provides an introduction to Tivoli Service Desk administration in a variety of business environments. It outlines the tasks for which the TSD administrator is responsible. It provides instructions for performing these tasks or indicates where you can find these instructions in the TSD on-line help.

- *Tivoli Service Desk Installation Guide*

  This book provides step-by-step procedures for installing all applications in the Tivoli Service Desk suite. It also contains instructions for creating Tivoli Service Desk suite tables and views.

- *Tivoli Service Desk Networking Guide*

  This book is for network administrators assigned to support the installation and maintenance of Tivoli Service Desk applications, and system integrators assigned to support the integration of Tivoli Service Desk applications with the Tivoli Framework.

Document Conventions

Certain conventions are used in TSD books to distinguish important pieces of information. The conventions include the following:

- Typefaces
- Special elements
- Product name acronyms

Typeface conventions

Typeface conventions are used to call attention to different uses of text in the book.

Bold — Bold is used for information that you must type exactly as shown in any instructive text.

Example: Type **c:/dev.kb** at the prompt.

Bold is also used for GUI elements that are the object of an action.

Example: From the Desktop menu, choose **Install Product**.

Italics — Italics are used for new terms and values that you must provide.

Example: Type *your ID* in the ID box.

`Monospace` — Monospace fonts are used for code examples.

Example: `else if x > y`

Special elements

Special elements are used to help you locate information—presented as Notes and Cautions—to assist you as you use the book. These special elements are shown here.

Note: Notes are points that offer information and explain special circumstances.

CAUTION:
Cautions alert you when data integrity is at risk as the result of specific tasks.

Product name acronyms

Tivoli Systems' product names are often referred to with acronyms. The table that follows shows the most common use of product name acronyms.

Product Name	Acronym
Tivoli Service Desk	TSD
Tivoli Problem Management	TPM
Tivoli Asset Management	TAM
Tivoli Change Management	TCM
Tivoli Problem Management Network/System Management	TPM NSM

Contacting Customer Support

This section updates the customer support information provided in the Tivoli Service Desk 6.0 documentation.

For support for this or any Tivoli product, you can contact Tivoli Customer Support in one of the following ways:

- Visit our internet site at http://www.tivoli.com/support
- Send e-mail to support@tivoli.com

Customers in the United States can also call 1-800-TIVOLI8 (1-800-848-6548). International customers should consult the internet site for customer support telephone numbers. You can also review the Customer Support Handbook, which is available on our internet site at http://www.tivoli.com/support/handbook/.

When you contact Tivoli Customer Support, be prepared to provide identification information for your company so that support personnel can assist you more readily. Company identification information may also be needed to access various online services available on the internet site.

The internet site offers extensive information, including a guide to support services (the Customer Support Handbook); frequently asked questions (FAQs); and documentation for all Tivoli products, including Release Notes, Redbooks, and Whitepapers. The documentation sets for some product releases are available in both PDF and HTML formats. Translated documents are also available for some product releases.

You can order documentation by e-mail at swdist@tivoli.com. Please provide the part number or order number of the desired document; alternatively, you can provide the document's title, version number, and date of publication.

We are very interested in hearing about your experience with Tivoli products and documentation. We also welcome your suggestions for improvements. If you have comments or suggestions about our documentation, please contact us in one of the following ways:

- Send e-mail to pubs@tivoli.com
- Fill out our customer feedback survey at http://www.tivoli.com/support/feedback.

1

Introduction to Tivoli Problem Management End-User Web Interface

Welcome to TPM End-User Web Interface, an internet and intranet interface to your TPM system. TPM End-User Web Interface 6.001 enables web users to do the following:

- Query a TPM database for a solution to their request
- Send a request to the TPM support center if no solution exists
- Track the progress of their request

In most cases, TPM End-User Web Interface users can solve problems without assistance from your support center.

Architecture

This section describes the different components that comprise the TPM End-User Web Interface: the corporate web server, web communication layer, and Application Server.

- The corporate web server is the gateway for clients to interact with TPM End-User Web Interface. The HTML, CGI scripts, and hypermedia files should reside on your web server.
- The web communication layer is a portal between the perl CGI scripts and the Application Server. The web communication layer is launched as a program or daemon (service on Windows NT).

- The Application Server communicates to the TPM database and maintains the business logic. Every problem request is processed by the Application Server.

Note: Whenever you allow public internet users to access your company server, you should make sure your firewall or other network securities will protect your TPM data.

How TPM End-User Web Interface Works

Every user account, whether new or anonymous, includes rights that determine the request capabilities and diagnostics available to a user. The following sections describe a client logging on (Typical session) and the different tools the client can use to search for solutions in your TPM database (Using diagnostics).

Typical session

The following is only an example. A typical client session might vary greatly depending on how you have configured your installation of TPM End-User Web Interface.

1. A customer using an internet browser accesses your company's public homepage on the Internet.
2. The customer selects the TPM End-User Web Interface hyperlink.

   Note: You should customize your web page to rename this link, such as "Customer Service" or "Solving Your Problem."
3. The customer must either log on as an anonymous user or enter information about themselves so they can become a registered user.

   Note: You should determine how to differentiate the capabilities of both account types.
4. Once logged on, the customer can do the following:

   - Use diagnostics to search your TPM database for solutions to their request. (For more information on TPM diagnostics, see the next section, "Using diagnostics."

- Submit their request to the support center.
- View already submitted requests and associated solutions.

5. If the customer has submitted a request, he can monitor the progress of the request by re-visiting your TPM End-User Web Interface site.

Using diagnostics

After a user has categorized their problem, they should select a diagnostic method to search for solutions. The following table briefly describes the TPM diagnostic methods. (For more comprehensive descriptions, see the *Tivoli Service Desk Administrator's Guide*.

Diagnostic Method	Description
Match Description	This is the Adaptive Learning diagnostic aid in TPM. Returns a solution according to the SCIM.
Step-by-Step	This is the HyperTrees diagnostic aid in TPM.
Hot News	The date and time on the Application Server and the user's organization are used to find a solution match.
Frequently Asked Questions	This is the Common Problem diagnostic aid in TPM. The selected SCIM and user location are used to find a solution.
Error Messages	This method returns solutions to error messages from TPM according to the SCIM.

Note: You can use the End-User Web Interface Administration Notebook to prevent users from accessing some or all of the diagnostic aids. For more information on configuring diagnostic aids, see "Configuring TPM End-User Web Interface" on page 13.

2

Installing TPM End-User Web Interface

Implementing TPM End-User Web Interface is a process of configuring a component you already have (the Application Server) along with installing two new components (Perl scripts and web communication layer). To be sure you are ready to install, make sure the requirements in the following section are true.

Installation Requirements

Make sure the following requirements have been met before installing any components of TPM End-User Web Interface.

Application server installation requirements

Application server installation prerequisites include the following:

- Tivoli Service Desk patches 6.0-TSD-0001, 6.0-TSD-0004, and 6.0-TSD-0006
- TSDBuild installed and TSD tables built
- TSD files are parsed and working
- Database connectivity established

Web server installation requirements

Web server installation requirements include the following:

- CGI 1.1 compliant web server installed (such as Apache or Internet Information Server)

- Perl v5.002 or greater installed (also known as Microsoft ActivePerl for Windows)

  Note: For more information or for downloading Perl components, go to www.perl.com.

Web communication layer installation requirements

Web communication layer installation requirements include the following:

- Tivoli Service Desk patch 6.0-TSD-0006 installed on the machine

Note: E-mail sent to clients must pass through the Notification monitor. For more information, see the *Tivoli Service Desk Administrator's Guide*.

Installing TPM End-User Web Interface

Note: Before you install TPM End-User Web Interface, extract the files into a directory making sure to maintain the directory structure while extracting.

Increasing TCP/IP ports for Windows NT

Use the following procedure to allow Windows NT to allocate more TCP/IP ports for your applications.

Note: Please disregard if you have UNIX servers.

1. Launch a DOS window and type **regedit** to launch the Registry Editor.
2. From the directory tree, go to the following directory location: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet \Services\Tcpip\Parameters
3. From the Edit menu, select a new DWORD value.
4. Replace the default DWORD name with **MaxUserPort**.
5. Double-click the new DWORD value you just created.

   Result: The Edit DWORD Value dialog box appears.

6. In the Value data box, type **65534** and select the decimal radio button.

TPM file installation

Use the following procedure to install TPM files.

1. Copy the files from each of the product folders (for example, Tivtsd\tpm) over the files in the corresponding folder for your TSD 6.0 installation directory (for example, C:\TIVTSD\TPM).
2. Launch TSDBuild and, from the Actions menu, choose **Execute Script File**, then choose the **600WAMig.xxx** file (where xxx is the code for your database).
3. From the Actions menu, choose the **Initialize Product Profile** option, then choose the **600WAPro.ini** file.
4. Parse the TSD 6.0 application using the **parse -a** command from a command line, or the **Parse all files** option from the **Tivoli Service Desk 6.0/Parse Utility** program group.
5. From a command line at the TIVTSD\TPM directory, parse the 6.0-TSD-0007 files by entering the following:
   - **kp -b webadmin**
   - **kp -b webacces**

Web server file installation

Use the following procedure to install web server files:

1. Copy the files from the **WebServer\scripts** directory to the location where the web server scripts reside on your machine (for example, c:\inetpub\scripts or /apache/cgi-bin).
2. Copy the files from the **WebServer\html** directory to the **<root>/tpm** directory on your machine (for example, c:\inetpub\wwwroot\tpm or /apache/htdocs/tpm).
3. If a sub-directory called **graphics** does not already exist in the directory where the HTML files are located, create one.
4. Copy the **logo.gif** file to the graphics directory you just created.

Note: Whatever image you choose to be re-named **logo.gif** will appear at the top of every page in the TPM End-User Web Interface product.

5. Configure the **web_access.cfg** file in the scripts directory.

   Note: For more information on configuring the web_access.cfg file, see "Configuring the CGI Scripts" on page 25.

6. Add the following link to your Intranet/Internet Home Page:

```
<A HREF="/scripts/web_access.cgi?FUNC=DISP_LOGIN">
TPM End-User Web Interface</A>
```

Applying a Perl script map

The following procedure is to associate a Perl script file for a Windows environment.

Note: Please disregard if you have UNIX servers.

1. Launch a DOS window and type **regedit** to launch the Registry Editor.
2. From the directory tree, go to the following directory location: HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\W3SVC\Parameters\ScriptMap
3. From the Edit menu, select a new string value.
4. Replace the default string name with **.pl**.
5. Go to the Edit menu again and create another new string value.
6. Replace the default string name with **.cgi**.
7. Double-click one of the new strings you just created.

   Result: The Edit String dialog box appears.
8. In the Value Data box, type the following:

   c:\perl\bin\perl.exe %s %s.
9. Repeat steps 7 and 8 to add value data for the other string.

Web communication file installation

The following are procedures to install the web communication layer.

For Windows NT:

1. From the .zip file, copy the files from the **webcomm\win32** directory to any directory on your machine.
2. Copy the files **net110.dll** and **tsdc111.dll** from the **TDT\win32** directory into the **winase32** directory. (Typically, c:\tivtsd\winase32.)

   Note: net110 will overwrite your existing net110 file and tsdc111 is a new library file.
3. Configure the **webcomm.ini** file.

   Note: For more information on configuring the webcomm.ini file, see "Configuring the Web Communication Layer" on page 21.
4. Install as a service. (For more information, see the next section, "Using TPM End-User Web Interface as a Service or Daemon.")
5. Restart your machine.

For UNIX:

1. From the .tar file, copy the files from the **webcomm/<OS>** directory to any directory on your machine.
2. Copy the files **libnet110.<ext>** and **libtsdc111.<ext>** from the appropriate OS subdirectory to the proper **TIVTSD/<os>ase/lib** directory. (For example, for HP-UX operating system, you would copy libnet110.sl and libtsdc111.sl from the TDT/hpux directory from the .tar file to the TIVTSD/hpuxase/lib directory.)

   Note: libnet110.<ext> will overwrite your existing libnet110.<ext> file and libtsdc111.<ext> is a new library file.
3. Configure the **webcomm.ini** file.

   Note: For more information on configuring the webcomm.ini file, see "Configuring the Web Communication Layer" on page 21.

4. Install as a daemon. (For more information, see the next section, "Using TPM End-User Web Interface as a Service or Daemon.")
5. Restart your machine.

Procedures to Install and Uninstall

Use the procedure appropriate to your operating system in the following table to install TPM End-User Web Interface as a service or daemon.

For Windows NT:

1. Configure the webcomm.ini file.

   Note: For more information see the "Configuring the Web Communication Layer" on page 21.

2. Add the location of the **webcomm.ini** to the environment path.
3. From the command prompt, go to the directory where the **webman.exe** file resides, similar to the following:

   `c:/webcomm/webcomm.exe`

4. Type the following:

   webman.exe –install –prog″**<*dir*>webcomm.exe**″.

   Note: For more information on arguments for webman or webcomm, use the **–help** argument.

5. Restart your machine.

   Result: The web communication layer will run as a service when the computer is restarted.

For UNIX:

1. Configure the webcomm.ini file.

   Note: For more information see the "Configuring the Web Communication Layer" on page 21.

2. Create a shell script that sets the environment variables for TDT, then launches the webcomm process. (See the section "Webcomm Shell Script Example" for more information.)
3. Add the shell script to the **/etc/inittab** file so it will respawn when the **webcomm** task recycles. (See your UNIX platform manpages for more information about the inittab file.)

   Note: For more information on arguments for webcomm, use the **–help** argument.
4. Restart your machine.

   Result: The web communication layer will run as a daemon when the computer is restarted.

Webcomm Shell Script Example

The following is a sample shell file called **startwebcomm.sh** for UNIX *only*.

```
#!/bin/sh

# Set environment variables

. /TIVTSD/.tsd_env_sh
PATH=/TIVTSD/wcomm:$PATH
export PATH

# Start webcomm

/TIVTSD/wcomm/webcomm
```

Note: For this example, the following line was added to the **etc/inittab** file:

```
webc:3:respawn:/TIVTSD/startwebcomm.sh
```

Procedure to uninstall

Use the procedure appropriate to your operating system in the following table to uninstall TPM End-User Web Interface as a service or daemon.

Operating System	Procedure
Windows NT	1. From the command prompt, go to the directory where the **webman.exe** file resides. 2. Type **webman.exe –remove**. **Result**: The web communication layer is removed as a service.
UNIX	Use the **kill <pid>** command to end the process. (A new webcomm process will be restarted by the init process unless the webcomm respawn command is removed.)

3

Configuring TPM End-User Web Interface

After installing TPM End-User Web Interface, you should customize the application to fit the needs of your company. The following are the three customization areas:

- Defining values for new and anonymous accounts using the End-User Web Interface Administration Notebook (described in this chapter).
- Defining values for the CGI scripts (found in “Configuring the CGI Scripts” on page 25).
- Defining values for the web communication layer (found in “Configuring the Web Communication Layer” on page 21).

Before you can define values for new and anonymous accounts using the End-User Web Interface Administration Notebook, you must first add End-User Web Administration Rights to your profile. The following section describes this process.

Adding End-User Web Administration Rights to Your Profile

After launching Tivoli Service Desk, use the following procedure to set values for new accounts:

1. Launch the Profile Editor.
2. Choose the **System Actions** tab.

3. Using the scroll bar, choose **End-User Web Administration**.
4. Type the following system action information in the appropriate boxes (if it does not already appear):
   - Name: **End-User Web Administration**
   - Action: **webadmin:WebConfigNotebook()**
   - Description: **End-User Web Administration**
   - Rights: **EWEB SYSTEM ADMINISTRATION**

   Note: These values are only present if you have run **600WAPro.ini** from TSDBuild. (For more information, see the section "TPM file installation" on page 7). Otherwise, the action will appear as **ewadmin**.
5. Choose the **Add** button below the **System Action** group box.
6. Choose the **Action Group** tab.
7. Using the scroll bar, choose the **Web Administration** action group.
8. Type the following action group information in the appropriate boxes (if it does not already appear):
   - Name: **Web Administration**
   - File: **webadmin**
9. Choose the **Add** button below the **Action Group** group box.
10. Choose the **Profiles** tab.
11. Add the system action **End-User Web Administration**.
12. Choose **OK**.

    Note: If you need more information on using the Profile Editor, see the *Tivoli Service Desk Administrator's Guide*.

Operation of the End-User Web Administration Notebook

You must define the following settings to properly configure TPM End-User Web Interface at your site:

- Request and security
- Account (new and anonymous accounts)
- Notification

The following sections cover these tasks.

Configuring request and security settings

To successfully create a request in TPM using TPM End-User Web Interface, more information is required than simply submitting a problem description. You must define the following request and security areas:

- Login security
- How the system is displayed to registered and anonymous users
- Default values when a problem is submitted

After launching the End-User Web Administration Notebook, use the following procedure to set values for requests and security:

1. Choose the **Request/Security Settings** tab.

   Result: The Request/Security Settings tab appears.

2. From the **Login Security** area, choose which type or set of users will be allowed to use TPM End-User Web Interface. Values are the following:

   - Registered Users Only
   - Registered Users and Create New Accounts
   - Registered Users and Anonymous Users
   - Anonymous Users Only

3. Below the **Login Security** area, select whether to **Display System to Registered User** or **Display System to Anonymous User**.

   Note: You can also select both values.

4. Use the following table to define values in the **Use these values when creating a problem** area.

Problem Record Value	Description
Call Code	The type of interaction. The default is Web Interaction.
Problem Code	The initial state of the TPM End-User Web Interface requests. States usually refer to whether the state of the problem is Open, Closed, or Transferred. (The default is Open.)
Severity	The severity of the problems submitted by TPM End-User Web Interface. The Severity list contains the severity number (with the lowest number indicating the greater importance) and a severity name or description. The default is 3 - Medium.
System	The value in this field is used for all requests submitted by TPM End-User Web Interface. This mandatory field has a default of WEBUnclassified.
User ID	This Tivoli Service Desk user is recorded as creating all TPM End-User Web Interface requests. The default is EXAV.
Recipient Group	This Tivoli Problem Management group receives the request transfers that TPM End-User Web Interface generates (all TPM End-User Web Interface requests are transfers). The default is EWEB. **Note:** TPM considers requests forwarded by TPM End-User Web Interface as Transferred. However, TPM End-User Web Interface consider forwarded requests as Open as the description for Problem Code states.

Configuring account settings

Account settings are used to specify different request and capability values for new and anonymous accounts. Categories for setting new account values are the following:

- Request capabilities
- Diagnostic capabilities
- Whether new accounts have immediate capabilities on registering
- Assigned knowledge rights

Categories for setting anonymous account values are the following:

- Request capabilities
- Diagnostics
- Assigned knowledge rights

The following sections are procedures to help you define values for new and anonymous accounts.

Procedure to set new account values

After launching the End-User Web Administration Notebook, use the following procedure to set default values for new accounts.

1. From the Account Settings tab, choose the **New Accounts** tab.

   Result: The New Accounts tab appears.

2. From the **Requests** area, choose what capability new accounts can use to create, view, and update problem requests, for example, if they will only be able to view their own requests and not others.
3. From the **Diagnostics** area, choose which methods new accounts can use to find a solution in the Tivoli Service Desk database.

   Note: If you need more information on the different diagnostic types, see the *Tivoli Service Desk Administrator's Guide*.

4. Enable the **Diagnostics only until reviewed** check box if you want to review registration information before giving users new account rights.
5. From the **Knowledge Rights** area, use the **Add** and **Remove** buttons to assign rights to new accounts.

Note: If you need more information on the different knowledge rights, see the *Tivoli Service Desk Administrator's Guide*.

Procedure to set anonymous account values

After launching the End-User Web Administration Notebook, use the following procedure to set values for new accounts.

1. From the **Account Settings** tab, choose the **Anonymous Account** tab.

   Result: The Anonymous Account tab appears.

2. From the **Requests** area, choose what capability anonymous accounts can use to create, view, and update problem requests, for example, if they will only be able to view their own requests and not others.

3. From the **Diagnostics** area, choose which methods anonymous accounts can use to find a solution in the Tivoli Service Desk database.

   Note: If you need more information on the different diagnostic types, see the *Tivoli Service Desk Administrator's Guide*.

4. From the **Knowledge Rights** area, use the **Add** and **Remove** buttons to assign rights to anonymous accounts.

   Note: If you need more information on the different knowledge rights, see the *Tivoli Service Desk Administrator's Guide*.

Configuring notification settings

After you have determined the capabilities for new and anonymous accounts, you must specify if they should be notified by e-mail when certain events take place with their submitted problem requests.

Notification events for new and anonymous accounts are Create, Update, and Closed. Create is when the problem request is submitted. Update is when the status of a problem request has changed. Closed is when the submitted problem has been resolved.

After launching the End-User Web Administration Notebook, use the following procedure to set notification values:

1. From the Account Settings tab, choose the **Notification Settings** tab.

   Result: The Notification Settings tab appears.

2. From the **Registered User - Notify On** area, choose what events will cause a registered user (or new account) to be notified.

3. From the **Anonymous User - Notify On** area, choose what events will cause anonymous user (or anonymous account) to be notified.

4. From the **E-mail Notification Type** combo box, choose the e-mail application that your site will use to notify users.

   Note: The default value is **Internet Mail** because it is application-neutral.

4

Configuring the Web Communication Layer

This chapter describes the contents of the webcomm.ini file and a description of the different arguments and values contained in the file. After reviewing the argument and value definitions, you should be able to successfully configure the webcomm.ini file to meet the needs of your company.

Sample webcomm.ini File

Although the arguments in this file are correct (such as **ListenPort** and **TSDLogin**) the values (such as **2345** for ListenPort) are examples only.

```
[WebComm]
ListenPort=2345
TSDLogin=EXAV
TSDPassword=
LogFile=
PrintLogToScreen=0
UseCache=1
CallerTimeout=900
NumberOfAppServers=1

[AppServer1]
AppServHost=APP_SERVER_NAME_1
AppServPort=5000

[AppServer2]
AppServHost=APP_SERVER_NAME_2
AppServPort=5000
```

```
[AppServer3]
AppServHost=APP_SERVER_NAME_3
AppServPort=5000

[AppServer4]
AppServHost=APP_SERVER_NAME_4
AppServPort=5000

[AppServer5]
AppServHost=APP_SERVER_NAME_5
AppServPort=5000
```

Defining the webcomm.ini File

Use the following table to configure the webcomm.ini file.

Argument	Description
ListenPort	The port that the perl CGI scripts use to communicate. This value must be the same as the $PORT value in the WEB_ACCESS.CFG file.
TSDLogin	This is a user ID that can be used to login to the TSD product. If the user ID is invalid, the daemon will not communicate with the Application Server.
TSDPassword	The password for the TSDLogin ID. If the password is invalid, the daemon will not communicate with the Application Server.
LogFile	Enter a valid directory and file name (c:\logfile.log) to log messages that pass through the web communication layer. **Note:** The log file will grow until the entire disk space is used. Do not use this unless you are debugging the product and the administrator can maintain the log file size.
PrintLogToScreen	Set to **0** to disable the output to your computer screen. Set to **1** if you want to view debug messages on your computer screen. This will not work if the web communication layer is running as a daemon or service. To view the output, start the web communication layer with the parameters **–noservice** or **–nodaemon**. This will start the web communication layer as a normal program.

Argument	Description
UseCache	Set to **0** to do the following: ■ Disable caching of systems and problem codes in the web communication layer. ■ To support Knowledge Rights and Multiple Customer Support. Set to **1** if you want the web communication layer to cache a list of systems and problem codes. (This prevents a database scan with every client request).
CallerTimeout	Set to the number of seconds a client has between requests before they are logged out (for example, 900 seconds = 15 minutes).
NumberOfAppServers	Set to the number of Application Servers from which the web communication layer can request information. Information for each Application Server will have to be entered in the following fields.
AppServHost	The name or IP address of the Application Server. **Note:** Each Application Server must have its own name or IP address.
AppServPort	The port of the dispatcher for the Application Server. **Note:** Each Application Server must have its own port number.

Note: The webcomm.ini file has default arguments for five Application Servers, though more can be added. If only one Application Server is used, set **NumberOfAppServers** to **1** and define **[AppServer1]** with the name and port of the lone Application Server (remaining four are ignored).

5

Configuring the CGI Scripts

This chapter describes the contents of the web_access.cfg file and a description of the different arguments and values contained in the file. After reviewing the argument and value definitions, you should be able to successfully configure the web_access.cfg file to meet the needs of your company.

Sample web_access.cfg File

Although the arguments in this file are correct (such as **$HTML_PATH** and **$IMAGE_PATH**) the values (such as **c:/inetpub/wwwroot** for **$HTML_PATH**) are examples only.

Note: Though explanatory comments already appear in the web_access.cfg file, a comprehensive reference table describing the arguments and their values appears in the section “Defining the web_access.cfg File” on page 28.

```
# web_access.cfg
#
##########################################
#
# Path where the Web Server root directory resides
#
# Note: This directory is created when the Web Server is installed
#       'c:/inetpub/wwwroot' (IIS)
#       '/apache/htdocs' (apache)
#
# Use the / directory seperator for all platforms.  Do not use \!
#
##########################################
$ROOT_PATH = 'c:/inetpub/wwwroot';
```

```
##############################################
#
# Path where the HTML files reside
#
# Note: This directory will reside off the $ROOT_PATH set above
#
##############################################
$HTML_PATH = '/tpm';
##############################################
#
# Path where images reside
#
# Note:  This directory will reside off the $HTML_PATH set above
#
##############################################
$IMAGE_PATH = '/graphics';

##############################################
#
# Path where the Script files reside
#
##############################################
$SCRIPT_PATH = '/scripts';

##############################################
#
# Path where Hypertext media reside
#
# Note:  This directory will reside off the $HTML_PATH set above
#
##############################################
$MEDIA_PATH = '/media';

##############################################
#
# File extension for media graphic files
#
# Note: The file extension is used for all images located in the
#       $MEDIA_PATH above.  All images will have to be in this format.
#
##############################################
$MEDIA_IMAGE_EXTENSION = "gif";

##############################################
#
# Machine name where the Daemon resides.
#
# Note:  localhost can be used to represent this machine.
#
```

```
##############################################
$DAEMON = 'localhost';

##############################################
#
# Port number used to communicate with the web communication layer
#
# Note:  This has to match the port number specified in the webcomm.ini
#        file, or the Perl scripts and web communication layer will not be able to
#        communicate.
#
##############################################
$PORT = 2345;

##############################################
#
# Timeout for HTML page (in seconds).  0 means no timeout.
#
# Note:  For security purposes, the user will automatically be logged out
#        after inactivity for the amount of seconds displayed below.
#
##############################################
$HTML_TIMEOUT = 900;

##############################################
#
# Maximum number of Solution returned to the Client per page
#
# Note:  This is used to limit the number of Solutions displayed to
#        the user per page.
#
##############################################
$MAX_DIAG = 15;

##############################################
#
# Maximum number of Problems returned to the Client per page
#
# Note:  This is used to limit the number of Problems displayed to
#        the user per page.
#
##############################################
$MAX_PROB = 15;
 ##############################################
#
# Maximum number of Problems returned to the Client on the HomePage
#


# Note:  This is used to limit the number of Problems displayed
```

```
#        on the Home Page when the user log's in.
#
##############################################
$MAX_MRU_PROB = 5;

##############################################
#
# Log file settings
#
# NOTE:  The logging function is designed for debugging purposes only.
#        The Log File will grow and grow if not maintained by the Admin.
#
##############################################

$LOG_FILE_ON   = 0;                  # 0-Off 1-On
$LOG_FILE_NAME = 'c:/weblog';

##############################################
#
# Date configuration settings.  Use this to alter the date and time
# inputs and displays in the product.
#
# Note: These are case-sensitive.
#
##############################################
$MONTH_LETTER  = 'm';
$DAY_LETTER    = 'd';
$YEAR_LETTER   = 'y';
$DATE_INPUT_FORMAT   = 'mm-dd-yyyy';
$DATE_OUTPUT_FORMAT  = 'mm-dd-yyyy';

$DATE_TIME_SEPARATOR = ' at ';

$HOUR_LETTER   = 'H';
$MINUTE_LETTER = 'M';
$SECOND_LETTER = 'S';
$TIME_OUTPUT_FORMAT  = 'HH:MM:SS';
```

Defining the web_access.cfg File

Use the following table to configure the web_access.cfg file.

Argument	Description
$ROOT_PATH	Set to the path where the ROOT directory resides on the Web server (for example, **c:\inetpub\wwwroot** or **usr/local/apache/htdocs**).

Argument	Description
$HTML_PATH	Set to the directory where the HTML files reside on the Web server. Use the / directory separator for all operating systems (for example, /tpm). This directory will be added to the $ROOT_PATH to determine where the files are.
$IMAGE_PATH	Set to the directory where the IMAGE files reside on the Web server. Use the / directory separator for all operating systems. The **$IMAGE_PATH** will be added to the **$HTML_PATH** to locate any images used (for example, /graphics).
$SCRIPT_PATH	Set to the alias where the SCRIPT (CGI) files reside on the Web server. Use the / directory separator for all operating systems (for example, /scripts or /cgi-bin).
$MEDIA_PATH	Set to the alias where the MEDIA files reside on the Web server. Use the / directory separator for all operating systems (for example, /media).
$MEDIA_IMAGE_ EXTENSION	Set to the image format used for the media files (for example, .GIF or .JPG). **Note:** Most browsers only support the .GIF and .JPG image formats.
$DAEMON	Set to the name or IP address of the machine where the web communication layer resides.
$PORT	Set to the port number on which the web communication layer is listening. This value must be the same as the ListenPort value in the webdaemon.ini file.
$HTML_TIMEOUT	Set to the number of *seconds* a client has between requests before they are logged out (for example, 900 seconds = 15 minutes).
$MAX_DIAG	Set to the number of solutions that can be returned to the client for each request.
$MAX_PROB	Set to the number of problems that should be returned to the client for each request.

Argument	Description
$MAX_MRU_ PROB	Set to the number of problems that can be returned on the client Home Page. (The home page appears after the client logs in and displays the number of problems available.)
$LOG_FILE_ON	Set to **0** to disable the log file. Set to **1** to begin collecting activity information for the log file. **Note:** Do not use this unless you are debugging the product otherwise the log file will grow until the entire disk space is used.
$LOG_FILE_ NAME	Enter a valid directory and file name to log messages that pass through the CGI scripts (for example, c:\logfile.log).
$MONTH_ LETTER	Set a letter that represents the month value for local users (for example, *m* represents *month* in the U.S.).
$DAY_LETTER	Set a letter that represents the day value for local users (for example, *d* represents *day* in the U.S.).
$YEAR_LETTER	Set to a letter that represents the year value for local users (for example, *y* represents *year* in the U.S.).
$DATE_INPUT_ FORMAT	Set to the input string used for the date. This string must represent the letters configured in the $MONTH_LETTER, $DAY_LETTER, and $YEAR_LETTER variables (for example, *mm-dd-yyyy* represents the date in the U.S.).
$DATE_OUTPUT_ FORMAT	Set to the output string used to display the date. This string must represent the letters configured in the $MONTH_LETTER, $DAY_LETTER, and $YEAR_LETTER variables (for example, *mm-dd-yyyy* represents the date in the U.S.)
$DATE_TIME_ SEPARATOR	Set to a symbol or word that will separate the time and date when displayed (for example, **at**, which would create output of **01-01-2000 at 12:00:00** in the U.S.).

Argument	Description
$HOUR_LETTER	Set to a letter to represent the hour value for the local users (for example, *H* represents *hours* in the U.S.)
$MINUTE_ LETTER	Set to a letter to represent the minute value for local users (for example, *M* represents *minutes* in the U.S.).
$SECOND_ LETTER	Set to a letter to represent the seconds value for local users (for example, *S* represents *seconds* in the U.S.).
$TIME_OUTPUT_ FORMAT	Set the input string to display and enter the time. This string must represent the letters configured in the $HOUR_LETTER, $MINUTE_LETTER, and $SECOND_LETTER variables (for example, **HH:MM:SS** represents the time in the U.S.).

6

Troubleshooting

The following table has solutions to common problems you may have in configuring or maintaining TPM End-User Web Interface. For problems not covered here or if you need more assistance, see the section "Contacting Customer Support" on page xiii.

Problem	Possible Solution
Clients receive the following message when trying to submit a request: "Request Failed! There was an error processing the request"	Make sure the Application Server is running and that the **webcomm.ini** file is configured properly.
Clients receive the following error: "Internal Server Error 500"	TPM End-User Web Interface cannot read the perl scripts, which usually means that the scripts cannot locate the **perl.exe** file. Some web servers, such as Apache, require that the first line of your **web_access.cgi** have the path to the directory that contains the **perl.exe** file. If you find that the first line is already referencing the **perl.exe** file, view the web server error log file for more information.
Clients receive the following error: "Communication Failure! There was a communication error. Please, try again later."	Make sure the webdaemon is running and view the **web_access.cfg** and make sure it is configured properly.

Problem	Possible Solution
You cannot find the TPM End-User Web Administration Notebook to configure end-users.	After configuring TPM End-User Web Interface you can add an icon or menu item to your profile using the End-User Web Administration system action.
The web communication layer times out and fails to start.	Make sure that the **webcomm.ini** file is in the system environment's path.
Clients receive the following error: "This document contained no data. Try again later, or contact the server's administrator."	Make sure TDT 6.0.1 is installed on the machine(s) running the Application Server and web communication layer. Also, make sure the path to the web server root directory is correct in the **web_access.cfg** file.
You receive the following error on the machine running the Application Server: "ES-006 Error –13: Cannot send data back to the client".	Make sure TDT 6.0.1 is installed on the machine(s) running the Application Server and web communication layer.
When reparsing application you received the following message: ″Received String, expected Time.″	Start TSD Build and execute the single script p_view.cmt located in the TIVTSD\TSDBUILD\TPM directory.
Application Server fails to process web client transactions.	Agents for the Application Server have frozen or stopped communicating with the webcomm. Launch the Application Server Configuration notebook and change the Agent Threshhold value to 1500. Then, recycle (stop and restart) the application server. This will allow the agents to recycle more frequently. You can also prevent this by following the procedure in the section "Increasing TCP/IP ports for Windows NT" on page 6.

A

TPM Tables Used by TPM End-User Web Interface

This appendix is a reference of Tivoli Problem Management tables and fields used by TPM End-User Web Interface, as well as TPM End-User Web Interface tables and fields.

Certain tables in the TPM database need to remain in their original installation form to function correctly. For this reason, you should verify that the tables in this appendix have the same structure and fields as they did when you installed TPM.

CAUTION:
The following TPM tables may not function correctly if you alter them.

Tivoli Problem Management Tables

The following is a list of TPM tables, and their corresponding fields, used by TPM End-User Web Interface.

TPM Table	TPM Table Fields
CALL	CALL_ID CALL_BEGIN_DATE CALL_BEGIN_TIME CALL_END_DATE CALL_END_TIME USER_ID (FK) LOCATION_ID (FK) CALLER_ID CALLER_PHONE MODIFY_DATETIME
CALL_CODES	CALL_ONLY DESCRIPTION MODIFY_DATETIME
COMPONENT	SYSTEM_ID COMPONENT_ID

TPM Table	TPM Table Fields
CONTACTS	ACTIVE CONTACT CONTACT_ID CONTACT_NAME EMAIL_ADDRESS EMAIL_ADDRESS_TYPE EMAIL_ENABLED EMAIL_ID FLX_CON_DATE1 FLX_CON_DATE2 FLX_CON_INT1 FLX_CON_INT2 FLX_CON_INT3 FLX_CON_INT4 FLX_CON_TIME1 FLX_CON_TIME2 FLX_CON_VCHR1 FLX_CON_VCHR2 FLX_CON_VCHR3 FLX_CON_VCHR4 MODIFY_DATETIME PHONENUMBER
COUNTERS	COUNTER_NAME NEXT_NUMBER
DIAGNOSTIC_AID	AID_ID AID_TYPE AID_TITLE PARENT_ID IS_GROUP SOLUTION_ID SOLUTION_COUNT LOCATION_ID (FK) DIAG_NODE ACTION
DICTIONARY	WORD_ID WORD SYNONYM_ID

TPM Table	TPM Table Fields
GROUPS	GROUP_ID GROUP_NAME SECURITY_MASK MODIFY_DATETIME
HOMEPAGES	HOME_PAGE PAGETITLE
HOMESCREEN	HOME_PAGE POS OPERATION DATAVAL BMP
HYPERNODE	ACTION SYSTEM COMPONENT ITEM MODULE NODETITLE HYPERNODE_ID GOTONODE_ID (FK) QUESTN SOLUTION_ID (FK) SOLVED NODE_IN_PRODUCTION
ITEM	SYSTEM_ID COMPONENT_ID ITEM_ID
KEYWORD_MATRIX	PAGE_COUNT SOLUTION_ID WORD_ID

TPM Table	TPM Table Fields
LOCATION	ADDRESS CITY FAX_NUMBER FLX_LOC_DATE1 FLX_LOC_DATE2 FLX_LOC_INT1 FLX_LOC_INT2 FLX_LOC_INT3 FLX_LOC_INT4 FLX_LOC_TIME1 FLX_LOC_TIME2 FLX_LOC_VCHR1 FLX_LOC_VCHR2 FLX_LOC_VCHR3 FLX_LOC_VCHR4 LOCATION_ID LOCATION_NAME MODIFY_DATETIME PHONE_NUMBER STATE TIMEZONE_ID VISIBLE ZIP
MODULE	SYSTEM_ID COMPONENT_ID ITEM_ID MODULE_ID

TPM Table	TPM Table Fields
PROBLEMS	PROBLEM_ID SEVERITY (FK) USER_ID (FK) EMAIL_ADDRESS FIRST_CONTACT_ID (FK) FIRST_LOACTION_ID(FK) FIRST_SESSION_ID FIRST_CALL_ID TME_SPENT MODIFY_DATETIME PROBLEM_CODE (FK) OPEN_DATE OPEN_TIME PROBLEM_RESULT CLOSE_DATE CLOSE_TIME SOLUTION_ID
PROBLEM_HISTORY	PROBHIST_ID PROBLEM_ID (FK) USER_ID (FK) MODIFY_DATETIME ENTRY_DATE ENTRY_TIME ENTRY_TYPE ENTRY
PROB_TRANSFER	PROBLEM_ID (FK) GROUP_ID (FK)TRANSFER_DATE TRANSFER_TIME TRANSFERED_BY (FK) MODIFY_DATETIME
RESPONSE	REPONSE_TEXT HYPERNODE_ID (FK) CHILD_HYPERNODE

TPM Table	TPM Table Fields
SESSION	SESSION_ID SESSION_BEGIN_DATE SESSION_BEGIN_TIME SESSION_END_DATE SESSION_END_TIME CALL_CODE SEVERITY USER_ID CALL_ID (FK) PROBLEM_ID (FK) DESCRIPTION
SEVERITY_LEVELS	DESCRIPTION MODIFY_DATETIME SEVERITY_LEVEL
SOLUTIONS	USAGE_COUNT SYSTEM COMPONENT ITEM MODULE DESCRIPTION SOLUTION PROBLEM_TYPE (FK) SOLUTION_ID DESCRIPTION
SYSTEM	DESCRIPTION MODIFY_DATETIME SECURITY SYSTEM_ID

TPM Table	TPM Table Fields
USERS	ALARM_POLL_PERIOD FLX_USR_DATE1 FLX_USR_DATE2 FLX_USR_INT1 FLX_USR_INT2 FLX_USR_INT3 FLX_USR_INT4 FLX_USR_VCHR1 FLX_USR_VCHR2 FLX_USR_VCHR3 FLX_USR_VCHR4 FLX_USR_TIME1 FLX_USR_TIME2 MODIFY_DATETIME SECURITY_MASK USER_ACTIVE_FLAG USER_FULLNAME USER_ID USER_PASSWORD
WORK_HISTORY	WORK_ID WORK_BEGIN_DATE WORK_BEGIN_TIME WORK_END_DATE WORK_END_TIME DESCRIPTION

TPM End-User Web Interface Tables

The following is a list of TPM End-User Web Interface tables and their corresponding fields.

TPM End-User Web Table	TPM End-User Web Table Fields
EWEB_CONFIG	A_SYSTEM_FLAG A_COMPONENT_FLAG A_MODULE_FLAG CALL_CODE (FK) COMPONENT_FLAG GROUP_ID (FK) ITEM_FLAG LOGIN_SECURITY MODULE_FLAG NOTIFY PROBLEM_CODE (FK) SYSTEM_FLAG SYSTEM_ID (FK) SEVERITY_LEVEL (FK) USER_ID (FK)
EWEB_LOC_RIGHTS	CREATE_RIGHT DIAG_RIGHT KNOW_RIGHT LOCATION_ID (FK) UPDATE_RIGHT VIEW_RIGHT
EWEB_NEW_ACCOUNT	A_DIAG_RIGHT A_KNOW_RIGHT A_MAX_DIAG A_UNTIL_REV ACCT_MAX_DIAG CREATE_RIGHT DIAG_RIGHT EA_LOCATION_ID KNOW_RIGHT PREFIX UPDATE_RIGHT VIEW_RIGHT

TPM End-User Web Table	TPM End-User Web Table Fields
EWEB_USAGE	ADL_USAGE CP_USAGE DIAG_USAGE EM_USAGE HT_USAGE HN_USAGE SUBMIT_USAGE UPDATE_USAGE VIEW_USAGE WEB_ID (FK)
EWEB_USER_RIGHTS	A_UNTIL_REV CREATE_RIGHT DIAG_RIGHT MAX_DIAGNOSTICS UPDATE_RIGHT VIEW_RIGHT WEB_ID (FK)
EWEB_USERS	ACTIVE_FLAG EA_LOCATION_ID EA_CONTACT_ID KNOW_RIGHT REVIEWED PASSWORD WEB_ID (FK)
PROBLEM_CODES	DESCRIPTION MODIFY_DATETIME PROBLEM_CODE

B

Command Codes

This appendix contains reference tables for the command and subcommand protocol used in TPM End-User Web Interface.

Note: These codes are provided as reference only. Do not change them.

Reference tables for command and subcommands cover the following areas:

- CGI perl script to web communication layer subcommands
- Web communication layer to CGI perl script subcommands
- CGI perl script to Application Server command codes
- Web communication layer to CGI perl script command codes

Although each command and subcommand is one way, there is a corresponding command and subcommand going the other way. For example, the login subcommand sent from the perl script to the web communication layer ($CD_LOGIN_CMD) is a request to confirm a user logging into TPM End-User Web Interface. The corresponding login subcommand, going from the web communication layer to the perl script ($DC_LOGIN_RSP) is a responding subcommand that confirms a user logging into TPM End-User Web Interface.

CGI Perl Script to Web Communication Layer Subcommands

The following table is a reference of CGI perl script subcommands used to interact with the web communication layer.

Perl Script Subcommand	Value
$CD_LOGIN_CMD	0
$CD_VALID_USER_CMD	1
$CD_LOGOUT_CMD	2
$CD_APPSERVER_FUNCTION_CMD	3
$CD_APPSERVER_FUNCTION_NO_CALLER_CMD	4
$CD_ANONYMOUS_LOGIN_CMD	5
$CD_GET_SYSTEMS_CMD	6
$CD_GET_PROBLEM_CODES_CMD	7

Web Communication Layer to CGI Perl Script Subcommands

The following table is a reference of web communication layer subcommands to interact with the CGI perl script.

Web Communication Layer Subcommand	Value
$DC_LOGIN_RSP	0
$DC_VALID_USER_RSP	1
$DC_LOGOUT_RSP	2
$DC_APPSERVER_FUNCTION_RSP	3
$DC_APPSERVER_FUNCTION_NO_CALLER_RSP	4
$DC_ANONYMOUS_LOGIN_RSP	5
$DC_GET_SYSTEMS_RSP	6
$DC_GET_PROBLEM_CODES_RSP	7

CGI Perl Script to Application Server Command Codes

The following table is a reference of Perl script commands used to interact with the Application Server.

Command ID	Fields and Type
$CD_VALID_USERCMD	USER_AUTH_ID (Integer)
$CD_LOGOUT _CMD	USER_AUTH_ID (Integer)
$CD_LOGIN_CMD	Filename (webacces) Function (WA_Login) USER_ID (String) PASSWORD (String
$CD_ANONYMOUS _LOGIN_CMD	Filename (webacces) Function (WA_AnonymousLogin)
CreateNewUser	Filename (webacces) Function (WA_CreateNewUser) USER_ID (String) EMAIL_ADDRESS (String) FIRST_NAME (String) MIDDLE_NAME (String) LAST_NAME (String) PHONE_NUMBER (String)
GetLoginSecurity	Filename (webacces) Function (WA_GetLoginSecurity)
SendLoginInfo	Filename (webacces) Function (WA_SendLoginInfo) EMAIL_ADDRESS (String)
CreateProblem	Filename (webacces) Function (WA_CreateProblem) USER_AUTH_ID (String) SYSTEM (String) DESCRIPTION (String) EMAIL_ADDRESS (String) URL_PREFIX (String)

Command ID	Fields and Type
UpdateProblem	Filename (webacces) Function (WA_UpdateProblem) USER_AUTH_ ID (String) PROBLEM_ID (String) DESCRIPTION (String)
ProblemInquiry	Filename (webacces) Function (WA_ProblemInquiry) USER_AUTH_ID (String) VIEW_MASK (String) (1,2,3: own, org, all) PROBLEM_ID (String) SYSTEM (String) PROBLEM_CODE (String) BEGIN_DATE (MM/DD/YYYY) END_DATE (MM/DD/YYYY) START_INDEX (String) MAX_REQ (String)
ProblemDetail	Filename (webacces) Function (WA_ProblemDetail) PROBLEM_ID (String)
GetProblemHistory	Filename (webacces) Function (WA_GetProblemHistory) PROBLEM_ID (String)
GetWorkHistory	Filename (webacces) Function (WA_GetWorkHistory) PROBLEM_ID (String)
GetDiagAids	Filename (webacces) Function (WA_GetDiagAids) KNOW_MASK (String) DIAGNOSTIC_TYPE (String) SYSTEM (String) DESC'N/PARENT_ID (String) START_INDEX (String) MAX_DIAG (String)
GetHomePages	Filename (webacces) Function (WA_GetHomePages) KNOW_MASK (String) PAGE_ID (String)

Command ID	Fields and Type
SolutionDetail	Filename (webacces) Function (WA_SolutionDetail) KNOW_MASK (String) SOLUTION_ID (String)
HypernodeDetail	Filename (webacces) Function (WA_ HypernodeDetail) KNOW_MASK (String) HYPERNODE_ID (String)
GetCallerInfo	Filename (webacces) Function (WA_GetCallerInfo) USER_AUTH_ID (String)
ModifyUser	Filename (webacces) Function (WA_ModifyUser) USER_AUTH_ID (String) EMAIL_ADDRESS (String) FIRST_NAME (String) MIDDLE_NAME (String) LAST_NAME (String) PHONE (String)
ChangePassword	Filename (webacces) Function (WA_ ChangePassword) USER_AUTH_ID (String) OLD_PASSWORD (String) NEW_PASSWORD (String)
$CD_GET_SYSTEMS_ CMD	Filename (webacces) Function (WA_GetSystems) USER_AUTH_ID (String)
$CD_GET_PROBLEM_ CODES_CMD	Filename (webacces) Function (WA_ GetProblemCodes) USER_AUTH_ID (String)
GetSystemFlag	Filename (webacces) Function (WA_GetSystemFlag) USER_AUTH_ID (String)

Web Communication Layer to CGI Perl Script Command Codes

The following table is a reference of web communication layer commands used to interact with the Perl script.

Command ID	Fields	Value
$DC_VALID_USER_RSP	SUCCESS or FAIL (Integer)	1 or 0
$DC_LOGOUT_RSP	SUCCES or FAIL (Integer)	1 or 0
$DC_LOGIN_RSP	SUCCESS or ERROR (Integer) USER_AUTH_ID (Integer) CONTACT_ID (String) ORGANIZATION_ID (String) CREATE_RIGHT (Integer) VIEW_RIGHT (Integer) UPDATE_RIGHT (Integer) DIAG_RIGHT (Integer) KNOW_MASK (Integer)	1or <0 >0 Default Org. 0, 1, 2, or 3 0, 1, 2, or 3 Bit Mask Bit Mask
$DC_ANONYMOUS _LOGIN_RSP	SUCCESS or ERROR (Integer) USER_AUTH_ID (Integer) CONTACT_ID (String) ORGANIZATION_ID (String) A_CREATE_RIGHT (Integer) A_VIEW_RIGHT (Integer) A_UPDATE_RIGHT (Integer) A_DIAG_RIGHT (Integer) A_KNOW_MASK (Integer)	0 0 or 1 0, 1, 2, or 3 0, 1, 2, or 3 Bit Mask Bit Mask
CreateNewUser	SUCCESS or ERROR (Integer)	1 or <0
GetLoginSecurity	SECURITY or ERROR (Integer)	1, 2, 3, 4, or <0
SendLoginInfo	SUCCESS or ERROR (Integer)	1 or <0
CreateProblem	SUCCESS (Integer) or ERROR PROBLEM_ID (String)	1 or <0
UpdateProblem	SUCCESS or ERROR (Integer)	1 or <0
ProblemInquiry (Returns a list of the following 3 items. Only one of these.)	# IN LIST (Integer) PROBLEM_ID (String) PROBLEM_CODE (String) DESCRIPTION (String) TOTAL_MATCHES (Integer)	>= 0

Command ID	Fields	Value
ProblemDetail	SUCCESS OR ERROR (Integer) PROBLEM_ID (String) PROBLEM_CODE (String) CONTACT_NAME (String) ORGANIZATION (String) DATE_SUBMITTED (String-Date) TIME_SUBMITTED (String-Time) DATE_CLOSED (String-Date) TIME_CLOSED (String-Time) SYSTEM (String) DESCRIPTION (String) PROBLEM_RESULT (String)	1 or <0 YYYMMDD HH:MM:SS YYYYMMDD HH:MM:SS
GetProblemHistory (Returns a list of the following)	# IN LIST (Integer) BEGIN_DATE (String-Date) BEGIN_TIME (String-Time) END_DATE (String-Date) END_TIME (String-Time) DESCRIPTION (String)	>=0 YYYYMMDD HH:MM:SS YYYYMMDD HH:MM:SS
GetWorkHistory (Returns a list of the following)	# IN LIST (Integer) BEGIN_DATE (String-Date) BEGIN_TIME (String-Time) END_DATE (String-Date) END_TIME (String-Time) DESCRIPTION (String)	>=0 YYYYMMDD HH:MM:SS YYYYMMDD HH:MM:SS
GetDiagAids (Returns a list of the following. Only one of these.)	# IN LIST (Integer) IS_GROUP (Integer) SOL'N_ID/AID_ID (Integer) [RATING] (Integer) DESCRIPTION (String) TOTAL_MATCHES (Integer)	>=0 1 or 2 >=0
GetHomePages (Returns a list of the following)	# IN LIST (Integer) PAGETITLE (String) OPERATION (Integer) PAGE_ID/HYPER_ID (Integer) PAGE/NODE (Integer) TITLE (String)	>=0 1, 2, or 3

Command ID	Fields	Value
SolutionDetail	SUCCESS or ERROR (Integer) PROBLEM_TYPE (String) DATE_ADDED (String- Date) TIME_ADDED (String-Time) DESCRIPTION (String) SOLUTION (String)	1 or <0 YYYYMMDD HH:MM:SS
HypernodeDetail (Returns a list of the following)	SUCCESS or ERROR (Integer) NODETITLE (String) ACTION (String) SOLVED (if solved is 1, use same as solution detail) (String) [else...] (String) QUESTION (String) # IN LIST (Integer) CHILD_HYPERNODE (Integer) RESPONSE_TEXT (String)	1 or <0
GetCallerInfo	SUCCESS or ERROR (Integer) EMAIL_ADDRESS (String) FIRST_NAME (String) MIDDLE_NAME (String) LAST_NAME (String) PHONE_NUMBER (String)	1 or <0
ModifyUser	SUCCESS or ERROR (Integer)	1 or <0
ChangePassword	SUCCESS or ERROR (Integer)	1 or <0
$DC_GET_SYSTEMS _RSP (Returns a list of the following)	#IN LIST (Integer) SYSTEM (String)	>=0
$DC_GET_PROBLEM _CODES_RSP (Returns a list of the following)	# IN LIST (Integer) SYSTEM (String)	>=0
GetSystemFlag	SYSTEM_FLAG (Integer)	1 or 0

C

Configuring the End-User Web Administration Notebook

You can customize the End-User Web Administration Notebook by doing the following:

- Adding a flex field to output on the Solution Detail page
- Adding a flex field for input on the Problem Inquiry page
- Adding hypermedia, such as images or text files

To create any of the previous, you must do one or more of the following tasks:

- Add HTML code to account for the added field(s)
- Change the Perl CGI scripts to account for the added field(s)
- Change the Application Server code to account for the added field(s)
- Parse the Application Server code and restart the Application Server

The following sections are examples of the code changes necessary to add flex fields to the Solution Detail and Problem Inquiry pages.

Note: Code changes appear in bold.

Adding a Flex Field to the Solution Detail Page

The tasks necessary to add a flex field to the Solution Detail page are the following:

- Modify the HTML file to account for an additional field.
- Modify Application Server functions to populate this field with the correct information.

Modifying the HTML

The following HTML code example shows the addition of a flex field (FLX_SOL_VCHR1) to the sol_detail.htm file.

```
[...]
<!-- Solution information -->
<TABLE BORDER=0>
  <TR ALIGN=LEFT VALIGN=TOP>
    <TD><B><<TYPE>><<_STANDARD_SEPARATOR>></B></TD>
<TD><<PROBLEM_TYPE_VAR>></TD>
  </TR>
  <TR ALIGN=LEFT VALIGN=TOP>
    <TD><B><<CREATED>><<_STANDARD_SEPARATOR>></B></TD>
<TD><<DATE_TIME_ADDED_VAR>></TD>
  </TR>
</TABLE>
```

```
<TABLE BORDER="0">
  <TR><TD ALIGN="LEFT"><B>Flx_Vchr1 Label:
  </B></TD></TR>
  <TR><TD><<FLX_VCHR1_VAR>></TD></TR>
</TABLE>

<BR>
```

```
<TABLE BORDER="0">
  <TR><TD ALIGN="LEFT"><B><<DESCRIPTION>><<_STANDARD_SEPARATOR>>
</B></TD></TR>
  <TR><TD><<DESCRIPTION_VAR>></TD></TR>
</TABLE>
<BR>

<TABLE BORDER="0">
  <TR><TD ALIGN="LEFT"><B><<SOLUTION>><<_STANDARD_SEPARATOR>>
</B></TD></TR>
  <TR><TD><<SOLUTION_VAR>></TD></TR>
</TABLE>

[...]
```

Modifying the Perl scripts

The following are Perl script code examples that show modifications to the scripts SolutionDetail() and HypernodeDetail() in solve.pl. (Both scripts are associated with sol_detail.htm.)

```
sub SolutionDetail {
[...]
if($argList[0] == $WA_SUCCESS) {
    $argIdx = 1;
    $VAR{'PROBLEM_TYPE_VAR'} = $argList[$argIdx++];
    $dateTime = FormatDateTimeStr( $argList[$argIdx++], $argList[$argIdx++]);
    $VAR{'DATE_TIME_ADDED_VAR'} = $dateTime;
    $VAR{'DESCRIPTION_VAR'} = FormatString( $argList[$argIdx++] );
    $VAR{'SOLUTION_VAR'} = FormatString( $argList[$argIdx++] );
    $VAR{'SOLUTION_TITLE_VAR'} = $SOLUTION;

 $VAR{'FLX_VCHR1_VAR'} = $argList[$argIdx++];
set_sol_detail_strings();
    EchoFile($HTML_PATH, "sol_detail.htm");
[...]
}

sub HypernodeDetail {
[...]
if ($solved ne 'TRUE') {
[...]
}
else {
    # This is a solved node.  Show the details of the solution.
    $VAR{'PROBLEM_TYPE_VAR'} = $argList[$argIdx++];
    $dateTime = FormatDateTimeStr( $argList[$argIdx++], $argList[$argIdx++]);
    $VAR{'DATE_TIME_ADDED_VAR'} = $dateTime;
    $VAR{'DESCRIPTION_VAR'} = FormatString( $argList[$argIdx++] );
    $VAR{'SOLUTION_VAR'} = FormatString( $argList[$argIdx++] );
    $VAR{'SOLUTION_TITLE_VAR'} = $VAR{'NODETITLE_VAR'};
    $VAR{'NODE_INFORMATION_VAR'} = $_NODE_INFORMATION_TAG;

$VAR{'FLX_VCHR1_VAR'} = $argList[$argIdx++];
set_sol_detail_strings();
    EchoFile($HTML_PATH, "sol_detail.htm");
}
[...]
}
}
```

Modifying the Application Server

The following are Application Server code examples that show modifications to the functions WA_SolutionDetail() and WA_HypernodeDetail() by adding FLX_SOL_VCHR1 : STRING; to the RequestRec structure.

```
PROCEDURE WA_SolutionDetail( REF MetaData, Data : LIST OF STRING ) IS
TYPES
    RequestRec IS RECORD
        problem_type    : STRING;
        add_date        : DATE;
        add_time        : TIME;
        description     : STRING;
        solution        : STRING;
        security        : INTEGER;

flx_sol_vchr1   : STRING;
    END;
[...]
```

Change the SQLSelectInto statement to also select FLX_SOL_VCHR1 from the solutions table. This will cause the RequestRec structure to contain the correct data.

```
--------------------
Get the solution
--------------------
oldFormat := (DateFormat($FmtDateISOFormat));
ret := SQLSelectInto('SELECT PROBLEM_TYPE, ADD_DATE, ADD_TIME, ' &
                     'DESCRIPTION, SOLUTION, SECURITY,
                      FLX_SOL_VCHR1 FROM SOLUTIONS
' & 'WHERE SOLUTION_ID = ?', solution_id,
request);
```

After the last AddParm() statement, add the following:

```
AddParm(MetaData, Data, '', request.problem_type);
AddParm(MetaData, Data, '', request.add_date);
AddParm(MetaData, Data, '', request.add_time);
AddParm(MetaData, Data, '', request.description);
AddParm(MetaData, Data, '', request.solution);
AddParm(MetaData, Data, '', request.flx_sol_vchr1);
```

The following modifications to WA_HypernodeDetail() are similar to modifications to WA_SolutionDetail, but without the reference to RequestRec structure.

```
PROCEDURE WA_HypernodeDetail( REF MetaData, Data : LIST OF STRING ) IS
[...]
VARIABLES
    ret          : INTEGER;
    know_mask    : INTEGER;
    oldFormat    : INTEGER; (* Used for reading dates correctly. *)
    hypernode_id : INTEGER;
    count        : INTEGER;
    security     : INTEGER;
    add_date     : DATE;
    add_time     : TIME;

flx_sol_vchr1: STRING;
cursor       : SQLCursor;
    hypernode    : Hypernode_Type;
    response     : ResponseRec;
    responseList : LIST OF ResponseRec;
ACTIONS
[...]
    IF (HyperNode.solved) THEN
        SQLSelectInto('SELECT ADD_DATE, ADD_TIME, SECURITY, FLX_SOL_VCHR1
               FROM SOLUTIONS ' &
                      'WHERE SOLUTION_ID = ?',
                      Hypernode.solution_id,
                      add_date, add_time, security, flx_sol_vchr1);
        [...]
        AddParm(MetaData, Data, '', HyperNode.description);
        AddParm(MetaData, Data, '', HyperNode.solution);

AddParm(MetaData, Data, '', flx_sol_vchr1);
    ELSE
       [...]
    END;

    DateFormat(oldFormat);
END;
```

After you have made these modifications, do the following:

1. From an Application Server command line, type **kp webacces**.
2. Restart the Application Server.

Adding a Flex Field to the Problem Inquiry Page

The tasks necessary to add a flex field to the Problem Inquiry page are the following:

- Modify the prob_inquiry.htm file to add a flex field with the label ″Call Code.″
- Modify the perl scripts ProblemInquiry(), DisplyProblemInquiry(), and WA_ProblemInquiry()

Modifying the HTML

The following HTML code example shows the addition of a call code field (CALL_CODE) to the prob_inquiry.htm file.

```
[...]
<!-- System, Problem Code choice boxes -->
    <TABLE BORDER="0">
        <TR><TD><B><<SYSTEM>><<_STANDARD_SEPARATOR>></B></TD>
        <TD><SELECT NAME="SYSTEM">
        <OPTION VALUE=""><<_SYSTEM_BOX_DEFAULT_VALUE>></OPTION>
        <<SYSTEM_LIST_VAR>>
        </SELECT></TD>
        </TR>

        <TR><TD><B><<STATUS>><<_STANDARD_SEPARATOR>></B></TD>
        <TD><SELECT NAME="PROBLEM_CODE">
        <OPTION VALUE=""><<_PROBLEMCODE_BOX_DEFAULT_VALUE>></OPTION>
        <<PROB_CODE_LIST_VAR>>
        </SELECT></TD>
        </TR>
<TR><TD><B>Call Code:</B></TD>
        <TD><INPUT NAME="CALL_CODE" SIZE="15"VALUE="<<CALL_CODE_VAR>>">
        </TD></TR>
</TABLE>
[...]
```

Modifying the Perl scripts

Modify ProblemInquiry() in view.pl as follows:

```
sub ProblemInquiry {
[...]
    if( exists($FORM{'START_IDX'})) {
        $startIdx = $FORM{'START_IDX'};
    }
    else {
        $startIdx = 1;
    }
    &AddArg($startIdx, $STRING);
    &AddArg($MAX_PROB, $STRING);

&AddArg($FORM{'CALL_CODE'}, $STRING);
```

```
&SendMsg($CGI_TO_DAEMON, $CD_APPSERVER_FUNCTION_CMD);
[...]
}
```

Also, modify DisplayProblemInquiry() so if the search does not return any problems, the call code field will contain the previous value.

```
sub DisplayProblemInquiry {
    ValidateUser();

    if((GetSystemList())&&(GetProbCodeList())) {
        $VAR{'PROBLEM_ID_VAR'} = $FORM{'PROBLEM_ID'};
        $VAR{'BEGIN_DATE_VAR'} = $FORM{'BEGIN_DATE'};
        $VAR{'END_DATE_VAR'} = $FORM{'END_DATE'};
        $VAR{'DATE_FORMAT_VAR'} = $DATE_INPUT_FORMAT;


$VAR{'CALL_CODE_VAR'} = $FORM{'CALL_CODE'};
        if (($viewRight >= 2) && ($organizationID ne '')) {
[...]
}
```

Now, modify WA_ProblemInquiry() as follows:

```
PROCEDURE WA_ProblemInquiry( REF MetaData, Data : LIST OF STRING ) IS
[...]
VARIABLES
[...]
    system       : STRING;
    problem_code : STRING;


call_code    : STRING;
    begin_date   : STRING;
    end_date     : STRING;
[...]
ACTIONS
    ----------------------
    -- Get the arguments
    ----------------------
    [...]
    begin_date      := Data[10];
    end_date        := Data[11];
    startIdx        := {Data[12]} : INTEGER;
    maxProb         := {Data[13]} : INTEGER;
    call_code       := Data[14];

    SetUnknown( MetaData );
    SetUnknown( Data );
```

```
    ---------------------
    -- Create the query
    ---------------------
    [...]
    IF (Known(system) AND (system <> '')) THEN
        probView.system := system;
    END;
    IF (Known(problem_code) AND (problem_code <> '')) THEN
        probView.problem_code := problem_code;
    END;
IF (Known(call_code) AND (call_code <> '')) THEN
        probView.call_code := call_code;
    END;
    oldFormat := (DateFormat($FmtDateISOFormat));
```

After you have made these modifications, do the following:

1. From an Application Server command line, type **kp webacces**.
2. Restart the Application Server.

Changing Control Types

If you followed the instructions in the previous section on adding a flex field with the label ″Call Code″ to the Problem Inquiry page, you can use the instructions in this section to change the Call Code text box to a combo box with one of the following features:

- Pre-defined call codes
- Automatically populated call codes from your TPM database

Note: The only reason you would pre-define call codes in your combo box is if you do not expect your call codes to change. If you expect your call codes to change, you should automatically populate the combo box with call codes from your TPM database.

Modifying the HTML to show pre-defined call codes

The following HTML code example shows the Call Code text box being changed to a combo box with the pre-defined call codes of Web Interaction and Incoming VMail.

```
<TR><TD><B>Call Code:</B></TD>
    <TD><SELECT NAME="CALL_CODE">
    <OPTION VALUE="">** Select a Call Code**</OPTION>
    <OPTION VALUE="Web Interaction">Web Interaction</OPTION>
    <OPTION VALUE="Incoming VMail">Incoming VMail</OPTION>
    <!-- Add other options here -->
    </SELECT></TD>
    </TR>
```

Modifying the HTML to show automatically populated call codes

The following HTML code example shows the Call Code text box being changed to a combo box with an Application Server function that queries your TPM database for call codes.

Note: If you added the options from the previous example, you must now remove them.

```
<TR><TD><B>Call Code:</B></TD>
    <TD><SELECT NAME="CALL_CODE">
    <OPTION VALUE="">** Select a Call Code **</OPTION>
    <<CALL_CODE_LIST_VAR>>
    </SELECT></TD>
    </TR>
```

Modifying the Perl scripts

You must now create a function that invokes a function on the Application Server that will return values as options for the combo box.

Note: Before you create this function, you should view the functions GetSystemList() and GetProblemList() in misc.pl because they are very similar to the one you will create.

The following function, GetCallCodeList, is the function on the Application Server that you will invoke in the function you create.

```
sub GetCallCodeList {
    if (&CreateSocket($DAEMON, $PORT)) {
        &AddArg($KML_FILE, $STRING);
        &AddArg('WA_GetCallCodes', $STRING);
        &AddArg($userAuthID, $STRING);
        &SendMsg($CGI_TO_DAEMON, $CD_APPSERVER_FUNCTION_CMD);
        ($cmd, $subCmd, $errCode) = &RecvMsg();
        &CloseSocket();
        if(($cmd == $DAEMON_TO_CGI)&&
```

```
($subCmd == $DC_APPSERVER_FUNCTION_RSP)&&($errCode == 0)) {
            $argIdx = 1;
            $callCodesList = "";
            if ($argList[0] >= 0) {
                for ($i = 0; $i < $argList[0]; $i++) {
                    if ($argList[$argIdx] eq $FORM{'CALL_CODE'}) {
                        $callCodesList .=
"<OPTION SELECTED VALUE=\"$argList[$argIdx]\">$argList[$argIdx++]\n";
                    }
                    else {
                        $callCodesList .=
"<OPTION VALUE=\"$argList[$argIdx]\">$argList[$argIdx++]\n";
                    }
                }
                $VAR{'CALL_CODE_LIST_VAR'} = $callCodesList;
    return(1);
            }
            else {
                AppServerError( 'GetCallCodeList', $argList[0] );
            }
        }
        else {
            DaemonError( 'GetCallCodeList', $errCode );
        }
    }
    else {
        CommunicationError();
    }
return(0);}
```

Next, the function must be called. The following function, DisplayProblemInquiry can do this.

```
sub DisplayProblemInquiry {
    ValidateUser();

    if((GetSystemList())&&(GetProbCodeList())&&(GetCallCodeList())) {
        $VAR{'PROBLEM_ID_VAR'} = $FORM{'PROBLEM_ID'};
        $VAR{'BEGIN_DATE_VAR'} = $FORM{'BEGIN_DATE'};
        $VAR{'END_DATE_VAR'} = $FORM{'END_DATE'};
        $VAR{'DATE_FORMAT_VAR'} = $DATE_INPUT_FORMAT;

        if (($viewRight >= 2) && ($organizationID ne '')) {
            # Add this checkbox if the user has rights to view
            # problems in his/her organization.
            $VAR{'INCLUDE_ORG_CHECKBOX_VAR'} = $_INCLUDE_ORG_CHECKBOX_TAG;
            if ($FORM{'INCLUDE_ORG'}) {
                $VAR{'INCLUDE_ORG_VAR'} = 'CHECKED';
            }
        }
```

```
        set_prob_inquiry_strings();
        EchoFile($HTML_PATH, "prob_inquiry.htm");
    }
}
```

Modifying the Application Server

Now you must create a function and name it **WA_GetCallCodes** in webacces.kb. (The function p_svcs.kb keeps the call codes cached, so you can use this function as a model for creating WA_GetCallCodes.)

```
PROCEDURE WA_GetCallCodes( REF MetaData, Data : LIST OF STRING ) IS
VARIABLES
    codeList : LIST OF STRING;
ACTIONS
    SetUnknown( MetaData );
    SetUnknown( Data );

    ----------------------
    -- Find the Codes
    ----------------------
    codeList := QueryCallCodes;
    ListSort( codeList );
    ListInsert(Data, {ListLength(codeList)} : STRING);
    ListInsert(Data, codeList);
END;
```

Add the following line (a forward declaration) at the top of webacces.kb:

```
KNOWLEDGEBASE webacces;

[...]

ROUTINES

[...]

PROCEDURE WA_GetSystems( REF MetaData, Data : LIST OF STRING );
PROCEDURE WA_GetProblemCodes( REF MetaData, Data : LIST OF STRING );
PROCEDURE WA_GetSystemFlag( REF MetaData, Data : LIST OF STRING );
PROCEDURE WA_GetCallCodes( REF MetaData, Data : LIST OF STRING );
PRIVATE
USES
```

After you have made these modifications, do the following:

1. Parse the Application Server code.

2. Restart the Application Server.

Note: If there was not a function available, such as QueryCallCodes, you can write one (view the code in p_svcs.kb as an example). Follow the example of other functions in webacces.kb when possible.

Adding Hypermedia

Diagnostics in the End-User Web Administration Notebook can include images in .GIF and .JPG formats and text files with a .TXT extension. Also, hypermedia files must reside in the $MEDIA_PATH directory as defined in the web_access.cfg file located on your web server.

If your company is using Service Ware Knowledge Packs, these *.TX2 files will need to be converted to .TXT files (use the **copy *.TX2 *.TXT** command in DOS, or **cp *.TX2 *.TXT** command in UNIX).

Index

T

U

W

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.