

# GoodLink™

Wireless Enterprise Messaging and Data Access System

---

## GoodAccess Administrator's & Developer's Guide

Version 1.0

## Copyright, trademark and patent information.

©Good Technology, Inc. 2001-2004. All rights reserved. All use subject to license terms posted at [www.good.com/legaldocs](http://www.good.com/legaldocs). Good, Good Technology, the Good logo, Good-Info, GoodAccess, GoodControl, GoodLink Forms, GoodLink and Powered by Good are trademarks of Good Technology, Inc. VeriSign(R) is a registered trademark of VeriSign, Inc. All other trademarks and service marks contained herein are the property of their respective owners. For example, Microsoft, Windows, Windows NT, Exchange and Outlook are trademarks of Microsoft Corporation. RIM, Research in Motion, RIM 950, RIM 957, and BlackBerry are registered trademarks or trademarks of Research in Motion Limited. Mobitex is a trademark of the Swedish Telecommunications Administration that may be registered in some jurisdictions. Datalight is a registered trademark of Datalight, Inc. FlashFX(tm) is a trademark of Datalight, Inc. Cingular, Cingular Wireless, the Cingular Icon, Xpress Mail, and Xpress Mail with GoodLink are trademarks of Cingular Wireless, LLC. All rights reserved.

Some or all of the following notices may apply to portions of the software or documentation provided by Good Technology, Inc.: Outside In® Wireless Export © 2001 Stellent Chicago, Inc. All rights reserved. Copyright 1993-2001 Datalight, Inc., All Rights Reserved. U.S. Patent Office 5,860,082. Code written by John Halleck is used with his permission. This distribution contains executables of the Netscape® Security Service (NSS) and Netscape Portable Runtime (NSPR). You may obtain the source code for these files from [www.mozilla.org](http://www.mozilla.org), which source files are subject to the Mozilla Public License 1.1. Part of the software embedded in this product is eCos - Embedded Configurable Operating System, a trademark of Red Hat. Portions created by Red Hat are Copyright (C) 1998, 1999, 2000 Red Hat, Inc. (<http://www.redhat.com/>). All Rights Reserved. THE SOFTWARE IN THIS PRODUCT WAS IN PART PROVIDED BY RED HAT AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED BY RED HAT. IN NO EVENT SHALL RED HAT BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. You may obtain a copy of the source code of the eCos Original Code from <http://www.redhat.com>. You may obtain a copy of source code of Good Technology, Inc.'s Modifications that have been publicly released in Executable form by sending an email to [support@good.com](mailto:support@good.com). The source code of the eCos Original Code and Good Technology, Inc.'s Modifications are subject to the Red Hat eCos Public License Version 1.1 (copy available at <http://www.redhat.com/>).

Some or all of the following notices may also apply to portions of the software or documentation provided by Good Technology, Inc.: ScriptEase(tm) Javascript/ECMAScript interpreter developed by Nombas, Inc. All Rights Reserved. This product includes software developed by the Apache Software Foundation (<http://www.apache.org>). Copyright (c) 2000-2003, The Apache Software Foundation and/or Yves Piguet. All rights reserved. Neither the name of Yves Piguet nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0> Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License. Copyright (c)1999-2001 Dan Adler, 315 E72 St. NY, NY, 10021 USA. mailto: [danadler@rcn.com](mailto:danadler@rcn.com) All rights reserved. The Jetty Package is Copyright Mort Bay Consulting Pty. Ltd. (Australia) and

others. Individual files in this package may contain additional copyright notices. The javax.servlet packages are copyright Sun Microsystems Inc. Copyright (c) 1990-2003 Sleepycat Software. All rights reserved. You may obtain a copy of the source code for the DB software from <http://www.sleepycat.com>. You may obtain a copy of source code of Good Technology, Inc.'s Modifications that have been publicly released in Executable form by sending an email to support@good.com. Copyright ©1996-1999 Corporation for National Research Initiatives; All Rights Reserved. Copyright (c) 1995-2000 by the Hypersonic SQL Group. All rights reserved. Copyright (c) 2001-2002, The HSQL Development Group. All rights reserved. Copyright 2002 (C) Nathaniel G. Auvil. All Rights Reserved. Copyright (c) 1998-2000 World Wide Web Consortium (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved. Copyright (c) 2001 MX4J. All rights reserved. Copyright 1994-2004 Sun Microsystems, Inc. All Rights Reserved. Copyright 1999, 2000 Boris Fomitchev Copyright 1994 Hewlett-Packard Company Copyright 1996, 97 Silicon Graphics Computer Systems, Inc. Copyright 1997 Moscow Center for SPARC Technology.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL ANY COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Good Technology, Inc. may have patents or pending patent applications, trademarks, copyrights or other intellectual property rights covering this subject matter. The software and documentation do not give you any license to these patents, trademarks, copyrights, or other intellectual property rights except as expressly provided in any written license agreement from Good Technology, Inc. The software and documentation may be covered by one or more patents as set forth at <http://www.rim.net/patents> which have been licensed by Research in Motion, Ltd. ("RIM") to Good. RIM is not affiliated with, nor does RIM endorse the operability of, the products or services described herein. Such patent license should not be construed as exhausting RIM's rights to royalties or damages or other compensation or relief or the grant of any express or implied license: (a) in relation to customer's use of third party products (except to the extent that use of third party email applications arises as a direct result of the customer using Good's products or services or the customer uses a third party wireless personal digital assistant or network carrier services in conjunction with Good's products or services); or (b) where customer or the supplier of the wireless personal digital assistant or wireless network services asserts any intellectual property rights against RIM notwithstanding the terms of clause (a) above, and RIM has exercised its right to suspend all or a portion of the licenses granted to Good.

## **Disclaimer**

No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Good Technology, Inc. Information in this document is subject to change without notice. This publication could include technical inaccuracies or typographical errors. Good Technology may make improvements or changes in the products or the programs described in this publication at any time.

Good Technology, Inc.  
4250 Burton Drive  
Santa Clara, CA, 95054  
Tel. (408) 327-6000 Fax (408) 327-6001  
Web site: [www.good.com](http://www.good.com)  
Printed In USA.

**Be Good. Be Safe.**  
**Please do not use while driving or engaged in any other activity that requires your full attention.**

# Contents

---

## **Preface**

Audience Definition	xi
Organization	xii
Documentation Conventions	xiii
GoodAccess Documentation Set	xiii
For More Information	xiv

## **1 Overview 1**

Key Features and Benefits	4
System Architecture	8
GoodAccess Server	9
Handheld Software	10
GoodLink Integration	10
Security Infrastructure	11

## **2 Server Installation 13**

Installation Roadmap	13
Preparing to Install	14
Getting A License Key	15
System Requirements	16
Using a Proxy Server	17
Multiple Server Issues	20
Creating Installation Accounts	21

Installing GoodAccess Server	23
Viewing Installed Files	29
Viewing Installed Certificates	29
Customizing Server Installation	30
Starting the Service	31
Stopping GoodAccess Server	32
Uninstalling GoodAccess Server	32
<b>3 Managing GoodAccess Server</b>	<b>35</b>
About the GoodAccess Server Console	36
About Server Properties	37
Opening the GoodAccess Server Console	38
Main	40
Users	40
Traffic	47
Logging	49
Site Access	54
Push	57
Administration	57
Stopping GoodAccess Server	58
“Best Practices”	58
Backup	58
Recovering from a Disaster	59
<b>4 Preparing User Handhelds</b>	<b>61</b>
Preparation Considerations	62
Installing GoodAccess Software on Handhelds	62
Enabling User Accounts	63
Setting Up GoodAccess on Handhelds	63
Starting GoodAccess on Handhelds	65
Uninstalling GoodAccess Software on Handhelds	65

<b>5</b>	<b>Using Push Technology</b>	<b>67</b>
	About Push	67
	Push Messages on the Handheld	68
	Push Messaging Tools	68
	Process Overview	69
	Using the Console Interface for Push Messaging	70
	Viewing Push Messages in the Console	70
	Sending Push Messages from the Console	71
	Generating Push Reports from the Console	73
	Setting Push Administration Options	74
	Using the Command Line Interface for Push Messaging	75
	Setting Up Push Messaging	75
	Invoking Push Commands	76
	Sending Push Messages	77
	Cancelling Push Messages	80
	Checking Push Status	81
	Generating Push Reports	84
	Monitoring Notifications	86
	Setting Push Properties	87
	Activation and Logging Properties	87
	Configuring Hosts	87
	Properties for the Push Console Interface	88
	The Push Database	89
	Delivery and Result Notification States	90
<b>6</b>	<b>Developing and Extending Applications</b>	<b>93</b>
	About Transformations	93
	Bibliography of Key References	95
	Overview of the Development Process	96
	Picking Compatible Applications	97
	Setting Up the GoodAccess Properties File	97

Setting Up the Application Properties File	99
Using Built-In Transformations	100
Using Debugging Transformations	102
Using Custom Transformations	103
Testing and Debugging Applications	104
Log Support	104
Debugging Tips	105
Using Transformation Debugging Tools	106
Deploying GoodAccess Applications	113
Creating Custom Transformations – An Example	114
Analyzing the Site	114
Updating the Applications Properties File	116
Creating a Custom Style Sheet	117
Processing Result	119
Customizing Tips	119
<b>7 Server Configuration Parameters</b>	<b>121</b>
About Properties Files	122
The Default Properties File	123
Application Properties Files	126
Client Configuration	127
Content Conversion and Transformation	128
Content Conversion Adapters	128
Content Transformation	129
Host Substitution	130
URL Substitution	132
Redirected HTTP Requests	134
Managing Cookies	136
Header Management	137
Default Header-Management Properties	141
Differential Data	144
Response Size	148



Push Support	150
Server Console	150
Traffic Display	152
Log Management	153
Location of Log Files	153
Log Rolling	153
Time Format Used in Time Stamps	154
Configuring NT Event and Good Operations Logging	154
About the Access Log	156
SSL Support	157
Advanced Configuration	158
HTTP Request and Response Management	158
Outgoing Proxy Support	159
<b>8 Developer's Reference</b>	<b>165</b>
Application Support	165
Benefits of this Approach	166
Identifying the Application Requested	167
Selecting Pages and Providing Transformations	171
Selector Component Syntax	173
Selector Matching and Ranking	174
Ignoring Case	175
Best Practices for Application Properties	176
Supported HTML Elements	177
HTML Color Support	184
Unsupported HTML	185
Image support	185
Unsupported WML Features	187
Support for JavaScript	190
Using GoodInfo1.0 System Variables	199
<b>Index</b>	<b>201</b>



# Preface

---

Thank you for choosing GoodAccess™, a new addition to Good Technology's suite of wireless messaging and application access products.

This preface discusses the objectives, audience, and organization, and conventions used in this guide. It also describes how to get additional information about Good products and services.

## Audience Definition

This guide is written for system administrators who install and maintain GoodAccess Server and prepare GoodAccess handheld devices for users. It includes information for application developers who plan to customize corporate applications for GoodAccess.

To install GoodAccess, you should be familiar with:

- Microsoft® Windows Server 2000®
- Pocket PC handhelds running Windows Mobile 2003® software or Treo 600™ Smartphone handhelds (or other supported handhelds) running Palm OS® 4.1
- GoodLink™ Server, GoodLink Management Server, and GoodLink Management Console
- GoodLink applications

To develop GoodAccess applications, you should also be familiar with HTML, WML, XML, and XSLT standards. Good Technology also recommends (but does not require) you to be familiar with Java2 Platform Software and Java Server technology.

## Organization

This guide is organized into the following chapters:

- “Overview” – describes GoodAccess features and product components. It also includes a high-level overview of GoodAccess system architecture.
- “Server Installation” – describes information you need to gather before starting an installation. It also describes how to install GoodAccess Server.
- “Managing GoodAccess Server” – describes how to use GoodAccess Server Console to view and manage server features. It includes information on how to enable user accounts on the server.
- “Preparing User Handhelds” – describes how to prepare user handhelds for GoodAccess.
- “Using Push Technology” – describes how to use push technology to send messages and alerts to GoodAccess handhelds.
- “Developing and Extending Applications” – describes how application developers can prepare and integrate internal corporate applications for use with GoodAccess handhelds.
- “Server Configuration Parameters” – describes additional configuration options you can use to customize GoodAccess Server.
- “Developer’s Reference” – includes detailed information on GoodAccess system variables, HTML limitations, and other reference material useful for application developers.

## Documentation Conventions

This book uses the following conventions:

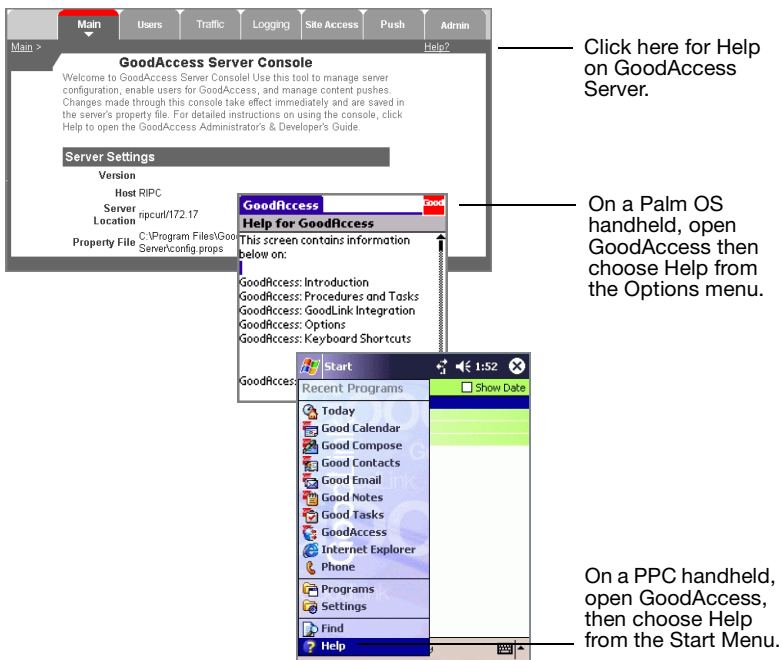
- *Italic text*  
Represents examples of text that appears on a screen or in a file. Also represents filenames, directories, and command variables.
- ***Boldfaced Italic text***  
Represents variables that are dependent upon your system configuration. For example, *GoodAccessAdmin*, indicates the account you use to run GoodAccess Server.
- **Bold text**  
Represents commands, buttons and other items that you select. For example, click **Next** to view the next screen in the GoodAccess installer.

## GoodAccess Documentation Set

The GoodAccess documentation set includes:

- *GoodAccess Administrator's and Developer's Guide* (this manual)  
Includes an overview of GoodAccess features and includes installation instructions for GoodAccess. Also includes information on managing GoodAccess Server and developing applications to work with GoodAccess.
- GoodAccess Server Help  
To access server Help, click the **Help** link in the top right corner of the GoodAccess Server Console.
- *GoodLink User's Guide*  
This book includes a chapter about using GoodAccess on a handheld.
- GoodAccess Handheld Help  
The Help system on your handheld includes a topic about using GoodAccess.
  - To access device Help on Palm OS handhelds, open GoodAccess and choose Help from the Options menu.

- To access device Help on the Pocket PC, open GoodAccess, then choose Help from the Start menu.



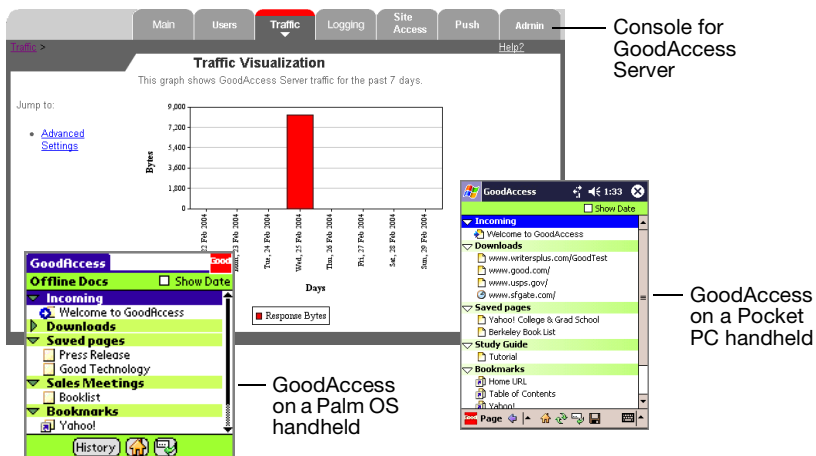
## For More Information

For more information about GoodAccess and other GoodLink products, visit the Good Technology Web site at <http://www.good.com>.

**Note:** Additional documentation from Good Technology is included on the GoodLink product CD.

# 1 Overview

Welcome to GoodAccess, the enterprise solution from Good Technology, Inc. that provides wireless access to back-end systems. The GoodAccess wireless system enables companies using GoodLink to extend valuable data sources – such as enterprise applications, intranets, and public Web sites – to mobile users.



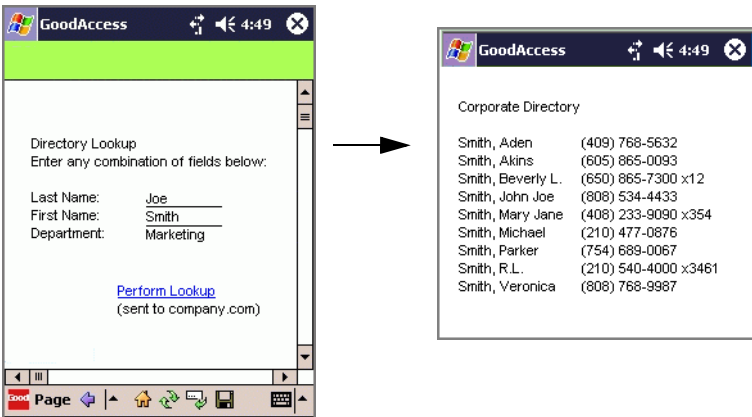
## Overview

Working in conjunction with your GoodLink Server and applications, the GoodAccess system includes:

- GoodAccess Server for accessing corporate and public Web-based data sources.
- GoodAccess software running on wireless handhelds for requesting and submitting data.

Unlike microbrowsers, GoodAccess is optimized for accessing data over today's wireless networks and doesn't require a continuous and synchronous connection. Mobile users can request or submit information quickly and easily by using GoodAccess applications on a handheld, and receive responses while working on other tasks. Users can also view information offline to maximize productivity.

An example of using GoodAccess to wirelessly access a corporate directory

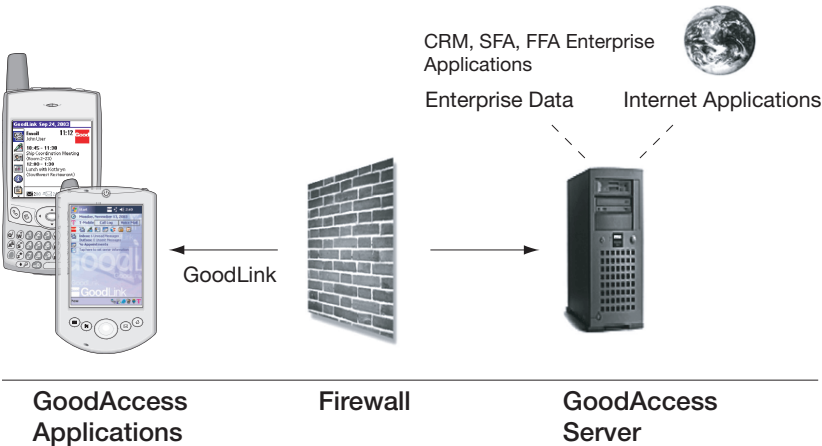


Mobile users request information with GoodAccess running on a wireless handheld.

GoodAccess delivers information from Web-enabled enterprise or public data sources to handhelds.



Corporate developers can use standard Web tools to adapt and extend existing applications for use with GoodAccess. GoodAccess is based on standard Web technologies, including XSLT, XML, CGI, and HTTP(S). GoodAccess applications can be easily distributed via GoodAccess transport to users' handhelds. Installation is simple and fast—users click one URL link to access the application and can begin using it immediately.



GoodAccess connects wireless handhelds to Web-enabled enterprise and public data

In addition to user-initiated data access, the GoodAccess system enables administrators to configure and proactively push messages, web pages, and other types of documents to user handhelds.

# Key Features and Benefits

The following sections summarize key features and benefits provided by GoodAccess.

## *Access to Back-End Systems from a Handheld*

- Displays existing HTML and WML pages on a handheld.
- XSLT transformations can filter and extract content to display relevant information only. The following transformations are provided out of the box:
  - Compressed HTML (to remove unnecessary content)
  - Tables replaced with bulleted lists
  - Images removed

Administrators can enable or disable transformations as desired or specify transformations for large pages only.

- Custom transformations can be developed to meet specific business processes and needs.
- GoodAccess can be integrated with back-end workflows to push alerts and information directly to the handheld. For example, technical bulletins can be re-broadcast automatically when content changes.

## *Optimized Delivery*

Delivery of information is optimized for slow networks with unreliable coverage:

- Web content is compressed to reduce page load time.
- For often-viewed pages, GoodAccess manages *differential data* and returns changes to pages, rather than reloading the entire page.
- Transformations eliminate the loading of unnecessary content.

***Enhanced Mobile User Experience***

- Taking advantage of background processing, users can multitask by downloading content while checking email, viewing saved information, or completing other tasks on the handheld.
- Users can store documents and other items offline, then view information while out of coverage. Information can be submitted into forms while offline; the content is automatically saved in the back-end system when the user connects to the network.
- Users can receive information on their handheld that is pushed immediately out to them or sent on a scheduled basis. Information is queued if a users is out of coverage.

***State-of-the-Art Security and Authentication***

- Messages are encrypted on GoodAccess Server (using triple-AES) and decrypted only when they reach the designated handheld. Encryption keys are stored on the customer's exchange server and on the device.
- Encryption keys are rotated on a configurable cycle time.
- SSL communications are provided for connections from GoodAccess Server to the Good Operations Center (also called GoodLink Data Center).
- Good Technology (and related suppliers) cannot read your enterprise traffic. The system cannot be wiretapped.
- A three-way authentication process occurs between GoodAccess Server, Good Operations Center, and handhelds.
- Handheld security includes:
  - Advanced password protection (SHA-1, a one-way hashing algorithm)
  - Remote handheld data protection. IT managers can wirelessly erase data from the handheld and terminate access.

### *GoodLink Integration and Compatibility*

- GoodAccess leverages the same security and transport layer as GoodLink.
- GoodLink and GoodAccess share installation and administration tools.
- Users familiar with the GoodLink interface can quickly learn to use GoodAccess.

### *Powerful System Management and Administration*

- With the *walled garden* feature, system administrators can limit user access to a specific set of sites and systems.
- System administrators can import user information for quick setup. User data can also be exported.
- To monitor handheld traffic volume, administrators can view global statistics for a particular GoodAccess Server.
- GoodAccess Server supports a watchdog service to monitor server status and proactively restart the server (if stopped). Administrators can configure the frequency of checks, the number of restarts, interval between restarts, and the notification email address.
- To support remote provisioning:
  - If GoodLink is installed, administrators do not need to re-cradle devices to install GoodAccess. Both cradled or non-cradled activation is supported.
  - No data is lost if power is lost on the device.
- Administrators can access a Web-based console for system to enable users, track handheld connection status, and so on.

***Reliability and Scalability***

- Redundant network architecture supports enterprise-class reliability. There is no single point of failure in the Good Operations Center and the infrastructure is self-healing.
- The Operations Center is designed for expansion and enterprise-class scalability. GoodAccess Server scales to support up to hundreds of users.
- Good Technology's Operations Center proactively monitors potential impacts to the data center. Traffic between Good servers and Operations Center is also monitored. Alerts are sent to the administrator if message flow is discontinued.

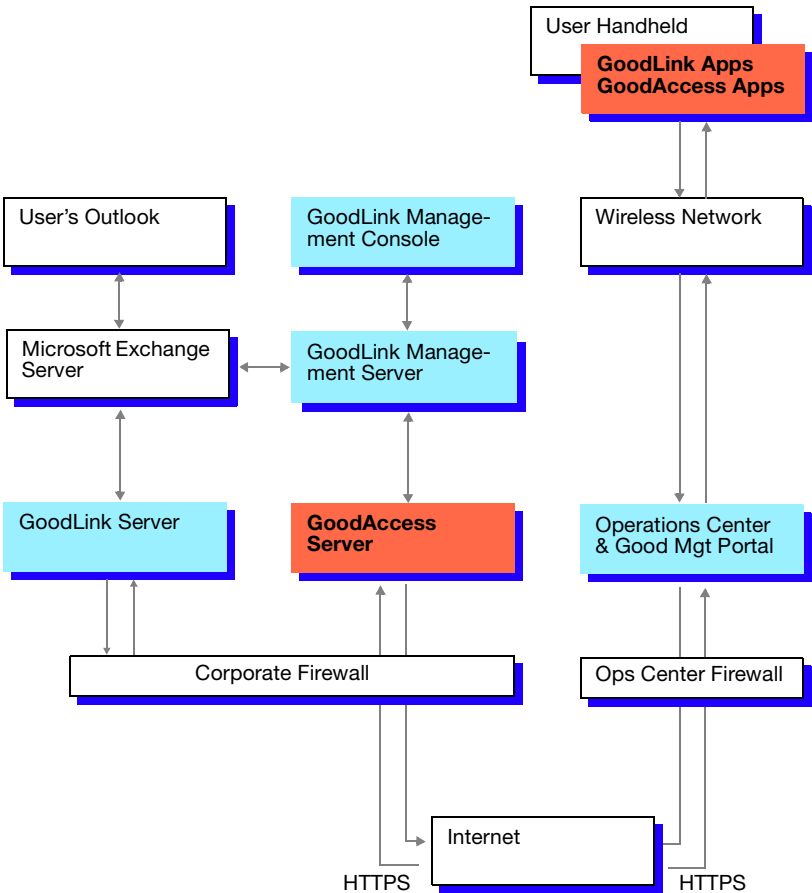
***Industry Standard Support***

The XSLT transformation engine leverages existing tools and standards for optimizing layout and content integration on small-screen devices. This includes:

- Content support for HTML, WML
- JavaScript and WMLScript scripting languages
- HTTP support, including Netscape-style Cookies, HTTP Basic Authentication, and standard Cache control directives
- XSLT-based transformation development and customization
- PAP protocol for push messaging

# System Architecture

The following illustration shows GoodAccess Server and GoodAccess handheld software integrated with your overall GoodLink network.

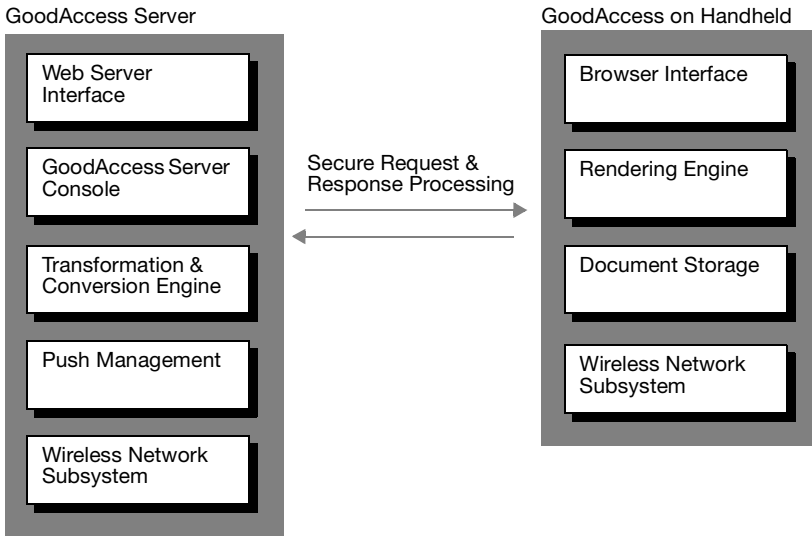


## GoodAccess Server

GoodAccess Server is built with Java Server technology. It includes:

- **GoodAccess Server Console**  
The server includes an HTTP-based management interface for server administration. You can access the console interface from any host on your network that runs Internet Explorer or from a GoodAccess handheld (if you have the appropriate access permissions).
- **Push Management**  
The server includes a Push Management system to support command line tools that can submit, cancel, and determine the status of pushed messages.
- **Wireless Network Subsystem**  
This server subsystem controls wireless transmissions between the server and the handheld. This subsystem communicates with the Good Data Operations Center to ensure continuous, reliable message delivery.
- **Logging Subsystem**  
The server supports an extensive logging facility you can use to view detailed server activity as well as troubleshoot problems that may occur.
- **Application Development Tools**  
The server includes an intelligent XML/XSLT transformation engine for graceful conversion of page content. You can develop your own applications to take advantage of these built-in transformations, or create transformations of your own.

## Overview



## Handheld Software

GoodAccess client software resides on the user handheld. It includes a rendering engine and browser interface for displaying pages and documents. Handheld software also includes a document storage system for storing and retrieving pushed documents.

## GoodLink Integration

GoodAccess is fully integrated with your existing GoodLink installation.

- GoodAccess handheld software is bundled with GoodLink 3.7 applications. It exists as a co-resident on the same wireless handheld.
- GoodAccess Server works with your existing GoodLink installation. Servers reside on the same corporate network, and communications are handled through the Good Operations Center.



## **Security Infrastructure**

Leveraging the GoodLink infrastructure, GoodAccess offers a complete, encrypted, end-to-end secure system.

First, a unique encryption key is generated and maintained for each user and stored on the handheld. Good Technology uses encryption technology standards to encrypt and decrypt all synchronizing messages and data exchanged between the user's corporate account and handheld. Second, data flows between GoodAccess Server and the Operations Center over the Internet using the HTTPS protocol. HTTPS provides secure communications through corporate firewalls using SSL. In this communication scheme, the Operations Center, which is identified by its URL, can be thought of as the server and GoodAccess Server as its client in a client/server relationship over the Internet.

Wireless key update enhances the security of the overall system by updating the key automatically every 30 days (configurable) from the date of handheld setup. The new key is sent to the handheld wirelessly and silently installed. Messages that are sent before the new key are decrypted with the prior key, and all messages that follow the new key are decrypted using it.



# 2 Server Installation

---

This chapter provides detailed instructions for installing GoodAccess Server.

**Important:** The instructions in this book assume you have already installed GoodLink Server, GoodLink Management Server, and GoodLink Management Console (version 3.7 or higher).

## Installation Roadmap

To get your users up and running, you will need to perform the following tasks. Each task is explained in detail in the following chapters.

- Review and perform pre-installation tasks.
- Install GoodAccess Server (includes GoodAccess Server Console).
- Start GoodAccess Server.
- Enable user accounts on GoodAccess Server.
- Prepare and activate user handhelds.

# Server Installation

**Important:** GoodAccess handheld software is bundled with GoodLink 3.7 or later. For convenience, you can install GoodAccess software on user handhelds before you install GoodAccess Server. Later, when GoodAccess Server is available, you can enable/activate GoodAccess software on the handhelds. Review the information in “Preparing User Handhelds” on page 61 before making a final decision.

## Preparing to Install

Before you install GoodAccess Server, there are some pre-installation tasks you must complete to set up installation accounts, host software, and so on. See the rest of the section for details. As you perform preinstallation tasks, use the following checklist to record your activities.

### *Installation Checklist*

Component	Information Requested	Values
GoodAccess Server	Host machine	_____
	Login Account &	_____
	Password	_____
	License Key	_____
	Installation Directory	_____
	Log Directory	_____
Good Operations Center	Cache Directory	_____
	URL Address	_____
	HTTP Proxy Server (if required)	_____
GoodLink Management Server	Host machine name	_____
GoodAccess Users	Handheld ID	Prepare a master list for each user who requires access to GoodAccess.
	Email address	
	GoodLink Account	[ ] Yes

## Getting A License Key

To install GoodAccess, you need a license key for the server.

In some cases, both serial number and license key are contained in email sent to you by your sales representative. Otherwise, follow this procedure to obtain the key:

1. Find and record the GoodAccess serial number and code number printed on a label on the GoodLink Server box and/or jewel case.
2. Go to <http://www.good.com/gsc> to obtain the license key for your GoodAccess Server.
  - a. Click the link to obtain a server license key.
  - b. Enter the GoodAccess serial number (s/n) and code from the label on the GoodLink Server box and/or jewel case into the appropriate fields.
  - c. Fill in the email address where you would like the server license key sent.
  - d. If you do not already have a Good Operations Center account, fill in the required information and select a password. The Good Operations Center allows you to monitor your handhelds and servers.

Once you've entered the necessary information, Good will register your GoodAccess Server. The server license key will be displayed at this time (only) in the Good Operations Center and it will be emailed to the email address you specify.

3. Record your serial number and license key on the installation checklist.

### System Requirements

Before you install GoodAccess Server software, make sure the host machine meets the following requirements.

Host requirements:

- Pentium III
- 800 MHz (minimum)
- 512 MB RAM required (1 GB recommended)
- 8 GB hard drive space free for GoodAccess Server
- Required minimum LAN speed for GoodAccess Server: 10 Mb/s.
- Windows 2000, SP4 or Windows 2003

Additional requirements:

- GoodAccess Server must be installed on a network running GoodLink Server, GoodLink Management Server, and GoodLink Management Console, Version 3.7 or higher.
- GoodLink Management Server should be in the same Windows domain as GoodAccess Server. Otherwise, a “trust” relationship must be created between the GMS domain and the domain GoodAccess Server runs in.
- If desired, you can install GoodAccess Server on the same host as GoodLink Server. However, for production environments, Good recommends installing GoodAccess Server on its own host. You cannot install GoodAccess Server on the same host as GoodLink Desktop or the GoodLink Management Console.
- You can install GoodAccess and GoodLink Forms (formerly called GoodInfo) on the same network. You do not have to uninstall GoodLink Forms to use GoodAccess – just make sure you don’t use the same server name for both.

**Important:** If you plan to use a combination of GoodLink Forms and GoodAccess applications, make sure your handheld devices and wireless carriers are supported by both GoodLink Forms and GoodAccess.

- The server host machine must have Internet access. They should be able to connect to http port 443 (secure https).

To check this, use a browser with proxy settings disabled on the host machine to connect to a secure remote location. You can use secure port 3101 or 4663 as an alternative to port 443.

You can also configure your firewall to allow communication between GoodAccess Server and a block of servers at the Good Operations Center in the IP range 216.136.156.65 to 216.136.156.96, inclusive.

- Before installing the server, ensure that the host machines' time and date are set to your network's correct time and date. Otherwise, errors such as a Security Alert regarding a problem with the site's security certificate may occur.
- The host for GoodAccess Server must be able to connect to the Web servers or application servers that GoodAccess applications are designed to access. To check the connection, use a browser on the GoodAccess Server host to connect to the Web servers or application servers.

## **Using a Proxy Server**

You can use an approved proxy server to communicate with Good Operations Center if you are unable to grant access via your firewall. The proxy server can be configured without granting additional access on the firewall.

There are several ways to set up proxy support for GoodAccess Server:

- Prior to installation (recommended method)
- During installation (if supported by the installer)
- After installation (recommended only for test installations)

### *Prior to Installation*

1. Before installing GoodAccess Server that is to be proxy-enabled, the following steps *must* be performed on the server host machine:
  - a. Right click on My Computer and select Properties.
  - b. Click the Advanced tab.
  - c. Click the Environment Variables button.
  - d. Under System Variables click New.
  - e. Enter *OverrideURL* as a new variable name.

If HTTP/1.1 Basic Authentication is required, enter:

*proxy://proxyuser:proxypasswd@proxyserver:proxyport/https://xml02.good.com/*

as the value, where:

*proxyuser* is the username to use with HTTP/1.1 Basic Authentication for authenticating to the Proxy.

*proxypasswd* is the password to use with HTTP/1.1 Basic Authentication for authenticating to the Proxy.

*proxyserver* is the DNS name of the proxy server to use.

*proxyport* is the port of the proxy server to use.

If HTTP/1.1 Basic Authentication is not required, enter

*proxy://proxyserver:proxyport/https://xml02.good.com/*

as the value, where:

*proxyserver* is the DNS name of the proxy server to use.

*proxyport* is the port of the proxy server to use.

2. Reboot the machine to activate the new setting.



### *During Installation*

During installation (if this option is available) you can enter the URL Address for the Good Operations Center that supports Good products installed on your network. For example:

*[http | https]://host:port/*

Where *host* is the Good Operations Center host and *port* is the port you use to connect to the host. You can connect using either an *http* or *https* connection.

To use a proxy server, enter the following instead:

*proxy://proxyuser:proxypassword@proxyserver:proxyport/[http | https]://host:port/*

where *proxyuser* and *proxypassword* are the login account and password for the proxy server, *proxyserver* is the host name (or IP address) of the proxy server, and *proxyport* is the port you use to connect to the proxy server.

The installer will create an *OverrideURL* environment variable that uses this connection information for the proxy server.

### *After Installation*

*Note:* This method is not recommended in a production environment.

If you set up a proxy server after installation, you must change the registry setting under *HKEY\_LOCAL\_MACHINE. SOFTWARE. Good Technology. GoodAccess Server. Host Network Address* = *proxy://proxyUser:proxyPassword@proxyHost:proxyPort/[http | https]://host:port/*

## Server Installation

### *Rules and Limitations*

The following rules and limitations apply if you want to use of a proxy server with GoodAccess Server:

- Uninstall does not remove or reset the *OverrideURL* environment variable. Ensure that it is updated appropriately before GoodAccess Server is installed.
- The proxy server must be configured to allow at least 5 minutes of idle time before timing out GoodAccess Server connections.
- The usernames and passwords for connecting to the proxy server must not contain ':', '@' or '/' characters.

See “Outgoing Proxy Support” on page 159 for more information on setting up proxy servers.

## Multiple Server Issues

**Important:** If you have more than one GoodAccess Server installed on your network, remember that:

- Each GoodAccess Server has an HTML interface (called the GoodAccess Server Console) you can use to enable users for GoodAccess. There is a separate console for each server host.
- Unlike GoodLink, users enabled on GoodAccess Server have account information stored with a specific GoodAccess Server.
- You cannot use the GoodAccess Server Console on one GoodAccess Server to enable users on another GoodAccess Server.
- To move a user to another GoodAccess Server (for example, in case of server failure), you must first disable the user account on the current server, then reenable the user account on the replacement server. See “Moving Users” on page 46 for more information.

- Any configuration changes you make with GoodAccess Server Console apply only to the server whose console you are using. These configuration settings are stored locally on the server.
- If you have more than one GoodAccess Server on the network, make sure their server configuration settings are compatible.
- Push messages for a particular user must be directed at the GoodAccess Server where the user is enabled.

## Creating Installation Accounts

GoodAccess Server runs as a service under a Windows NT user account that is set up for the server. In this guide, the example user account is called *GoodAccessAdmin*, but an existing account such as the *GoodAdmin* account can also be used.

To create a Windows NT account using Active Directory Users and Computers:

1. Using a Windows 2000 account with administrative privileges, log in to the host computer where you want to install GoodAccess Server.
2. Choose **Programs > Administrative Tools > Active Directory Users and Computers**.
3. In the Tree pane, select the Windows NT domain for the new user and choose **Users**.
4. Choose **New > User** from the **Action** drop-down menu.
5. Fill in the New Object - User dialog box.

As an example in this guide, the account for the GoodAccess Server user is *GoodAccessAdmin*. The name you use must be unique.

6. Click **Next**.

7. Enter and confirm a password for the new user.

The password is case sensitive. If selected, deselect the **User Must Change Password at Next Logon** option and any other selected options. Select the **Password Never Expires** option.

8. Click **Finish**.
9. Double-click *GoodAccessAdmin* in the user list or select *GoodAccessAdmin*, right-click, and choose **Properties** from the context menu.
10. In the *GoodAccessAdmin* Properties window, click the **Member Of** tab.
11. Using the **Add** button, go to the Select Groups window, select **Administrators** and click **Add**.
12. Click **OK** to close the Select Groups window. Click **OK** to close the Properties window.

*GoodAccessAdmin* is added to the appropriate group and given the necessary permissions.

13. Choose **Start > Programs > Administrative Tools > Local Security Policies**.
14. Expand **Local Security Policies** and select **User Rights Assignment**.
15. Confirm that the *GoodAccessAdmin* account has the **Log On As Service** right. If this right is missing, add it by double-clicking it and clicking **Add**, and then selecting the *GoodAccessAdmin* account and clicking **Add**:  
*Important:* If the **Log On As Service** right is missing, the installation or subsequent GoodAccess Server operations will fail.

## Installing GoodAccess Server

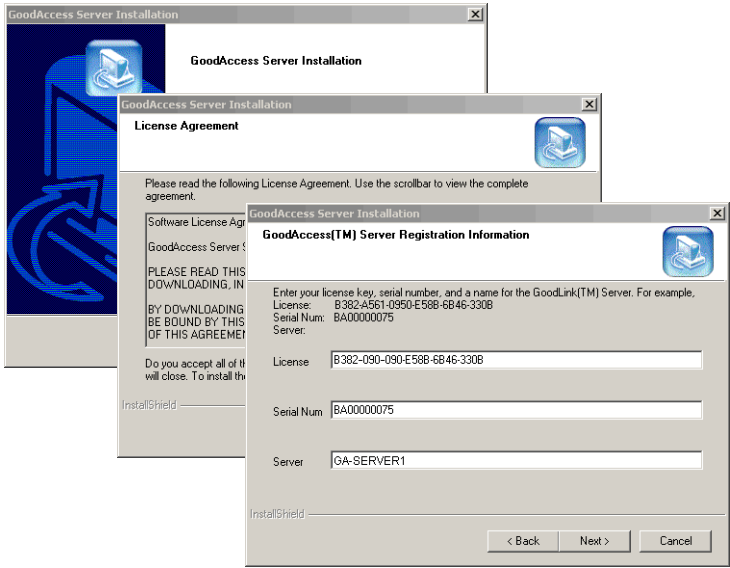
Use the following procedure to install a GoodAccess Server. Repeat the procedure for additional servers as needed. Each server can manage handhelds distributed across multiple GoodLink and Exchange servers. You can assign handhelds to GoodAccess Servers according to the organizational scheme most convenient to you.

1. Log in using the new administrative user account (*GoodAccessAdmin*) you created for GoodAccess Server.
2. Make sure the Services control panel is **closed**.
3. Insert the GoodLink CD. GoodAccess Server software is included on the GoodLink Server CD.
4. An autorun program will launch the installation program. If you don't use autorun, start *setup.exe* from the CD root.
5. A setup screen appears with a list of items you can install. Click **Add/Remove** for GoodAccess Server.

Installation files are extracted from the CD and an installation wizard appears.

6. Follow the installation instructions. Click **Next** to proceed to the next screen. Click **Back** to return to a previous screen.

# Server Installation



Installation information you specify includes:

## Installation Information

Item	Description
License Agreement	To proceed with the installation, you must accept the terms of the Good Technology software license agreement by clicking <b>Yes</b> .
Server Registration	Enter the GoodAccess serial number and site license key.
Server Name	Enter a descriptive name of your choice for GoodAccess Server. This name will appear in the GoodAccess Server Console. The name can be up to 16 characters long. No spaces allowed

*Installation Information*

---

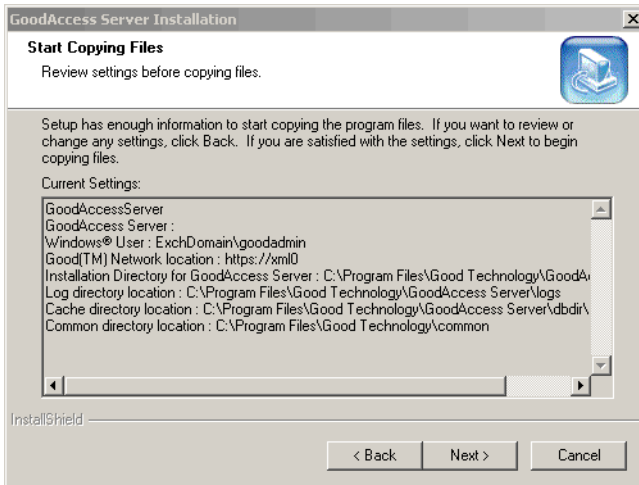
Item	Description
Good Network URL	<p>Enter the URL Address for the Good Operations Center that supports Good products installed on your network.</p> <p>After you enter this information, the installation program contacts the Operations Center, confirming the ability of the host to make the connection, and then validates the license key and serial number that you have provided.</p>
GoodLink Management Server Host	Enter the host name or IP address of the machine that runs the GoodLink Management Server for your network.
Server Location	Accept the default installation directory for GoodAccess Server software or browse to select a different location. If you select a directory that doesn't exist, the installer will create one for you.
Log Directory	Set the location of the directory to store server log files. You can store log files on the server host or on another host. If you select a directory that doesn't exist, the installer will create one for you.
Cache Directory	Set the location of the cache directory. This directory stores handheld data synchronization files created by the server. You can set the cache directory on the server host or on another host. If you select a directory that doesn't exist, the installer will create one for you.

### Installation Information

---

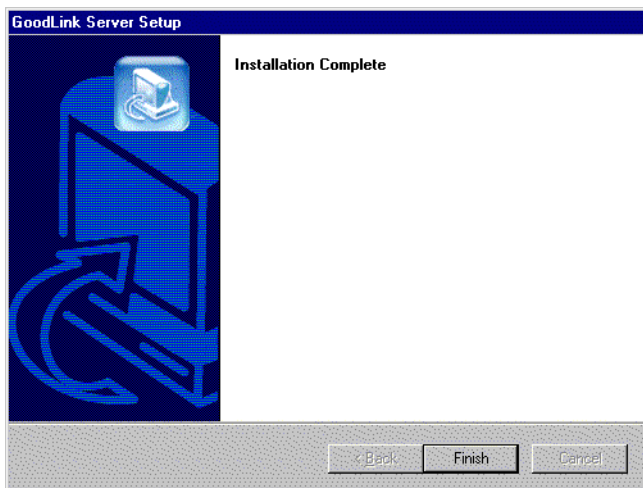
Item	Description
Windows NT Account Information	<p>Enter the Domain, Login Name, and Password for the Windows user account you're using to install GoodAccess Server. For example:</p> <p><i>Domain\GoodAccessAdmin</i> <i>opensesame</i></p> <p>This information is used to install GoodAccess Server as a service and start (or stop) the services.</p> <p><b>Note:</b> The domain and account names are not case sensitive, but the password is.</p>

When you have finished entering the required installation information, the installation wizard displays a summary screen of the information you have entered.



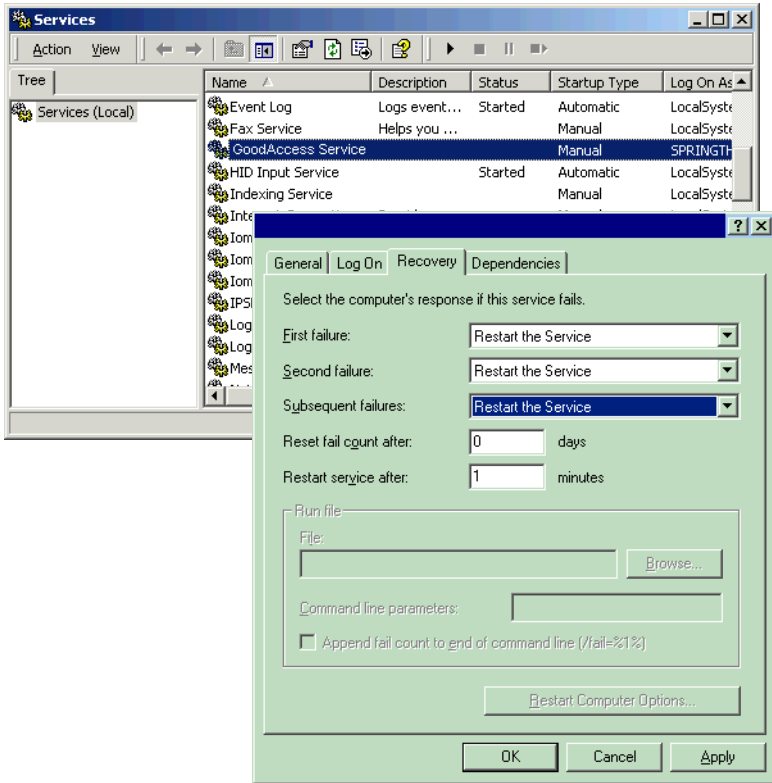


1. Check to make sure the information is correct, then click **Next**.  
When installation is complete, a prompt appears.



2. You can start the GoodAccess Server service now, or wait and customize server configuration, then start the service.  
See the following sections for more information.

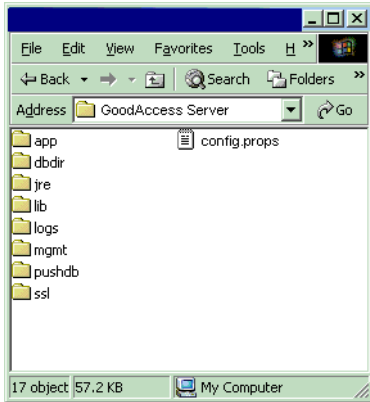
**Note:** During installation, GoodAccess Server is automatically configured to use the MS Windows service recovery feature. In case of unexpected failure, this feature will automatically attempt to restart the service.



If you want to view or change Recovery settings, open the Services control panel and select the **GoodAccess Service**. Then, choose **Properties > Recovery** from the context menu. (For more information about service recovery, see your Microsoft documentation).

## Viewing Installed Files

After you install GoodAccess Server, take a quick look at the installation directory. You should see the following directories and files.



## Viewing Installed Certificates

The Setup program automatically installs a certificate database that includes trusted root certificates for accessing sites with https protocol. To view the list of installed certification authority root certificates, see the following file in the installation folder where you installed GoodAccess Server:

*install\_dir/ssl/trusted.txt*

### Customizing Server Installation

GoodAccess Server is ready to start after installation. However, in some cases, you might wish to change a few configuration settings before starting the service (for example, if you are using a proxy server).

Server configuration is controlled by properties stored in a configuration file, called the GoodAccess Server properties file (*install\_dir/config.props*).

To edit the file:

1. Locate the file in the server installation directory.

By default, this file is located in *C:/Program Files/Good Technology/GoodAccess Server/config.props*

2. Make a backup copy of the file and set it aside for safe keeping.

A snapshot of the original server configuration (*config.props.backup*) is included in this directory. You can use this file to restore the default server properties. However, as you begin to customize your installation, it is a good idea to make backups as necessary to preserve the current contents of *config.props*.

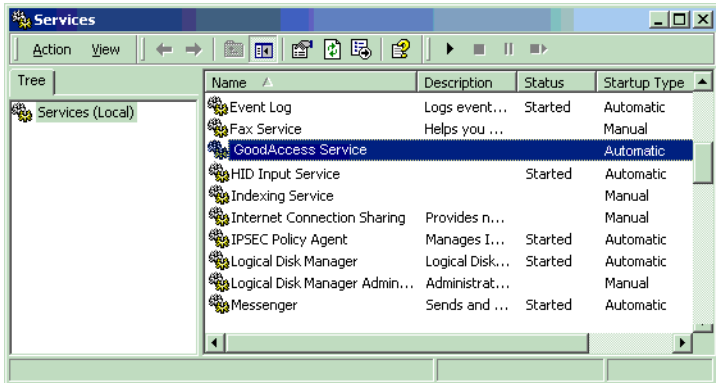
3. Open the file with a plain text editor (for example, Microsoft Notepad).
4. Change server configuration properties as desired.
5. Save your changes and exit the file.

For more information on changing server configuration, see “Managing GoodAccess Server” on page 35 and “Server Configuration Parameters” on page 121.

## Starting the Service

To start the GoodAccess Server service:

1. Open the Services panel.
2. Select **GoodAccess Server** and then click **Start**.



After starting the service, you can check that it is operating properly by:

- Opening the Windows NT Event Log. Information about successful and unsuccessful server actions appears here. For more information, see “NT Event Log and Good Operations Log” on page 52.
  - Using the GoodAccess Server Console to view server activities. For more information, see “Managing GoodAccess Server” on page 35.
3. Once the service is operating properly, you can enable user accounts on the server and prepare user handhelds for GoodAccess. See the following chapters for details.

# Stopping GoodAccess Server

Occasionally, you may need to stop GoodAccess services to reset server properties, load new software, or troubleshoot server problems.

*Important:* Stopping the server will interrupt user access. If you plan to stop the server for more than a few moments, make sure you reassign any handhelds managed by GoodAccess Server to a different server. For more information, see “Moving Users” on page 46.

To stop the GoodAccess Service:

1. Open the Services panel.
2. Select **GoodAccess Server** and then click **Stop**.

When you manually stop a service, the Windows automatic recovery facility does not try to automatically restart the server.

# Uninstalling GoodAccess Server

You may want to uninstall GoodAccess Server software from its host machine. For example, if you want to move the server to a different host or to install a later version of GoodAccess Server software.

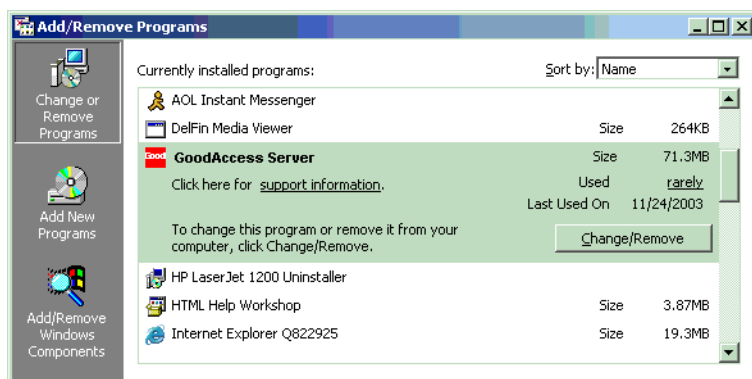
*Important:* In a production environment, make sure you reassign (either temporarily or permanently) any handhelds managed by GoodAccess Server to a different server before uninstalling. For more information, see “Moving Users” on page 46.

To uninstall GoodAccess Server:

1. Close all programs before proceeding with the uninstall.
2. Use the Services control panel to stop the GoodAccess Server service.

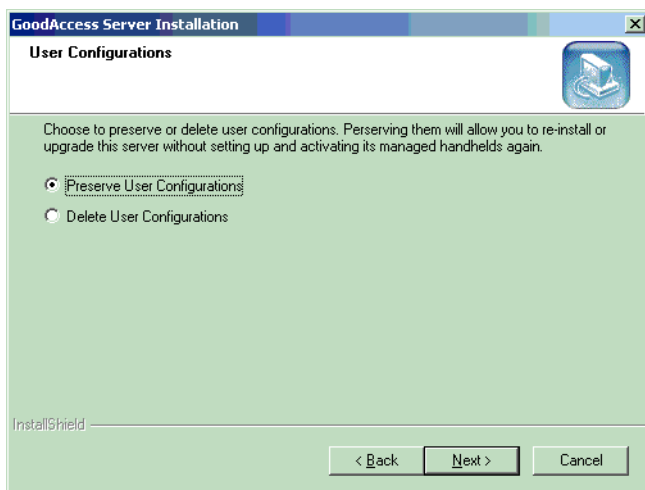
**Note:** When you manually stop GoodAccess Server, the Microsoft service recovery feature will not try to automatically restart the service.

3. After you stop the service, make sure you close the Services control panel.
4. To uninstall GoodAccess Server software from a host machine, go to the machine's **Control Panel** window and double-click **Add/Remove Programs**.
5. From the list of programs, select **GoodAccess Server**.
6. Click **Change/Remove**.



## Server Installation

7. Specify how you want to handle user configurations.



User configurations includes the list of users who have been enabled on the server.

**Important:** If you are uninstalling GoodAccess Server software to upgrade to a new version, make sure you select Preserve User Configurations.

8. Click **Next** to continue the uninstall.



# 3 Managing GoodAccess Server

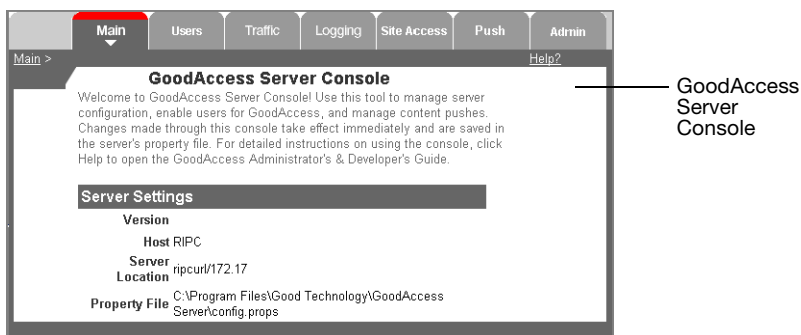
---

This chapter describes how to monitor GoodAccess Server to ensure server operations are occurring normally. It also describes how to manage user accounts on GoodAccess Server. Topics include:

- GoodAccess Server Console
- Managing User Accounts
- Monitoring Server Traffic
- Controlling Site Access
- GoodAccess Logging System
- Best Practices (redundancy, backup, and recovery)

# About the GoodAccess Server Console

GoodAccess Server includes an HTML interface, called the GoodAccess Server Console. You can use this interface to set server configuration, manage access for user handhelds, monitor network traffic, and so on. You can access the console from a standard Web browser (such as Microsoft Internet Explorer). You can also use GoodAccess software running on your handheld to access the console.



**Note:** The console includes software developed by the Apache Software Foundation (<http://www.apache.org>).

With the console, you can:

- Enable user access
- Monitor server traffic
- Configure Host and URL substitution
- Send push messages and alerts
- Set server logging properties

**Note:** Additional server configuration can be set in the GoodAccess Server properties file (*install\_dir/config.props*). See “About Server Properties” below for more information.

## About Server Properties

With few exceptions, the features available through the GoodAccess Server Console correspond to properties in the GoodAccess properties file, *config.props*.

Some server properties can be set either through the console or directly in the *config.props* file. However, there are important differences between the methods for setting properties:

- Properties set directly in the *config.props* file while the server is running do not take effect until the server is restarted.
- Properties changed through the console take effect immediately and are also saved in the *config.props* file.

**Important:** When you first install GoodAccess Server, the server makes a backup of the server configuration file and saves it as *config.props.bak*. You can use this file to restore the original server defaults if desired. Also, as you make changes to server configuration (using the GoodAccess Server Console interface or the *config.props* file), make sure you keep a current backup copy of the file.

## Opening the GoodAccess Server Console

Use the following procedure to open the console:

1. Open a Web browser and enter the following URL:

`http://hostname:port`

where **hostname** is the host name of GoodAccess Server and **port** is the port number.

The default port number 19001, but you can use the *admin.html.port* property in the *config.props* file to change this value.

A login screen appears.

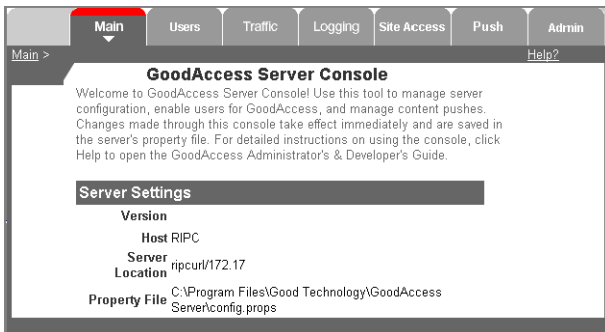
2. Enter the user name and password for the console and click **OK**. When you first login to the console, use the following values:

Login: *admin*

Password: *admin*

*Note:* Console login and password are defined by the *admin.html.username* and *admin.html.password* properties in the *config.props* file. To maintain system security, Good Technology recommends you change the default login and password. For more information, see “Administration” on page 57.

The server console appears.



3. Click the tabs to view status information and set attributes and configuration properties that are not on the Main page.

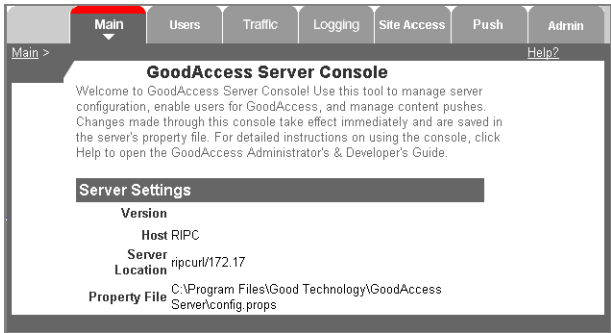
### *Server Console Pages*

---

Tab Name	Information Displayed
Main	Shows general information about the server including hostname and software version number.
Users	Shows enabled and disabled GoodAccess users. Use this page to enable access for user handhelds.
Traffic	Shows HTTP request and response traffic volume sent between wireless handhelds and the server. You can also use this page to configure traffic display options.
Logging	Enables or disables log files for common server components.
Site Access	Enables you to: <ul style="list-style-type: none"> <li>– Limit internet access (set Walled Garden) for users by disabling the <b>Go to Web Address</b> command on handhelds.</li> <li>– Set the default home page for handhelds.</li> <li>– Enable URL and hostname remapping for URL requests.</li> </ul>
Push	Enables you to send and manage push messages to users enabled on server. For more information, see “Using Push Technology” on page 67.
Admin	Enables you to manage login information for the GoodAccess Server. You can change the default administration login and password for the server and allow additional users to perform server administration.

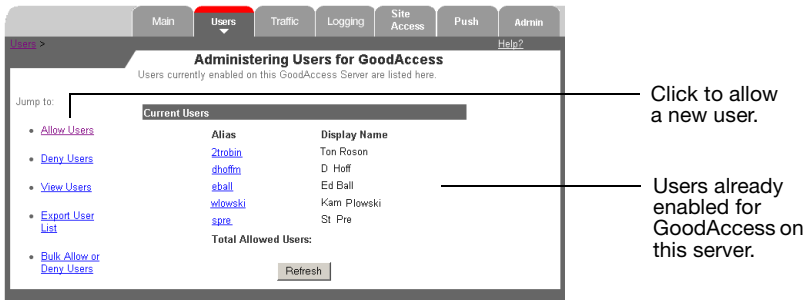
## Main

The Main tab of the server console displays general information about GoodAccess Server. This includes hostname, software version, and location of the server properties file (*config.props*).



## Users

The Users tab shows which user handhelds have access to GoodAccess services. When you first install and start the server, no users will appear on this page.



## Allowing a User

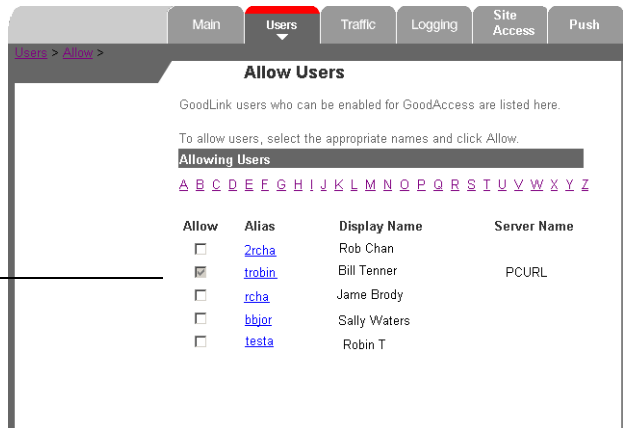
To allow a user:

1. In the server console, click **Users > Allow Users**.

An **Allow Users** page appears. This page contains a list of all users who have been provisioned on GoodLink.

2. Select the user or users you want to allow for GoodAccess and click **Allow**.

Select one or more users and click **Allow**.



3. When you have finished allowing users, click **Users** to return to the main **Users** page.

**Important:** If you have more than one GoodLink or GoodAccess server installed on your network, remember that:

- The Allow Users page shows all users on the network who have been provisioned on GoodLink, regardless of which GoodLink server was used for provisioning.
- The main Users page only shows users enabled on a specific GoodAccess Server (that is, the server listed on the Main tab of the console).
- You cannot use the server console on one GoodAccess Server to enable users on another GoodAccess Server.

### Viewing User Information

To view detailed information about users listed in the GoodAccess Server Console:

1. In the server console, click **Users > View Users**.

A page appears with detailed information about users. Both enabled and denied users are included. This information is useful for determining which users you want to allow for GoodAccess, troubleshooting access problems, and so on.



2. When you have finished viewing user information, click **Users** to return to the main **Users** page.



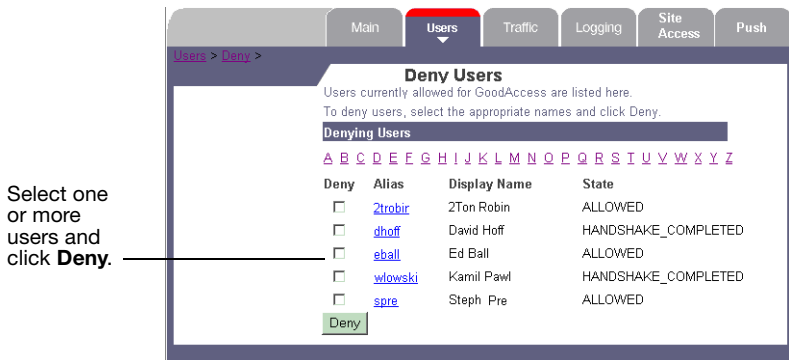
### Denying a User

Occasionally, you may want to deny access to a user who has been enabled for GoodAccess. To deny a user:

1. In the server console, click **Users** > **Deny Users**.

A **Deny Users** page appears. This page contains a list of users who are currently enabled on the server.

2. Select the user or users you want to deny access, then click **Deny**.



3. When you have finished denying users, click **Users** to return to the main **Users** page.

*Note:* When you deny a user, it may take a few minutes for the list of enabled users to be updated on the console.

### *Bulk Allowing or Denying Users*

If desired, you can allow or deny several users at a time by using information from a text file. To bulk allow or deny users:

1. Create a plain text file that includes a list of users. Create a separate line in the file for each user and include the following information:

*username, SMTP email address*

For example:

*Frank Jones, fjones@company.com*

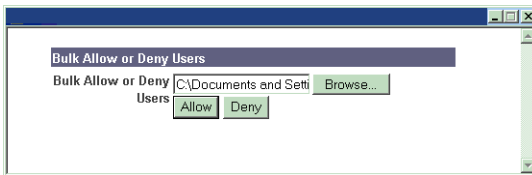
*Bill Martin, bmartin@company.com*

*Isaac Chen, ichen@hotlink.com*

2. In the server console, click **Users > Bulk Allow or Deny Users**.
3. Navigate to the file that contains the list of users.

You can enter a specific path to the file or click **Browse**.

- a. To enable access for users listed in the file, click **Allow**.
- b. To deny access, click **Deny**.



4. Click **Users** to return to the main **Users** page.

### *Exporting a List of Allowed Users*

You can also export a list of GoodAccess users to a plain text file. This information can be useful for preparing push scripts or creating lists of users to enable or deny.

To export a list of allowed users:

1. On the Management Control Panel, click **Users > Export Allowed Users**.

The server creates a plain text file (*users.txt*) that includes the following information for each GoodAccess user:

*email\_address, username, X400 domain\_name, device\_serial\_number*

For example:

```
"dhoff@myco.com", "dhoff", "/o= My Comp/ou=SF DEV/cn=Recipients/cn=dhoff",
"IMSI:31026060201"
"eball@myco.com", "eball", "/o=My Comp/ou=SF E2K/cn=Recipients/cn=eball",
"IMSI:60900424590"
"igo@hotmail.com", "meiko", "/o=NA/ou=SF Bay Area/cn=Recipients/cn=meiko",
"IMSI:310260424596"
```

A prompt appears asking you to open or save the file.

2. Click **Save** to save a copy of the file.

### *Moving Users*

In a multiple server site, if you move GoodAccess users from one server to another, it may take some time (up to three hours) for the second server to recognize that the user is available.

To immediately move a user to another server:

1. On the original server, deny access for the user. For details, see “Denying a User” on page 43.
2. Open the console on the new server.
3. Click **Users > Allow Users**.
4. Click the name of the user you want to add. The User Information page appears.
5. Click **Refresh**.

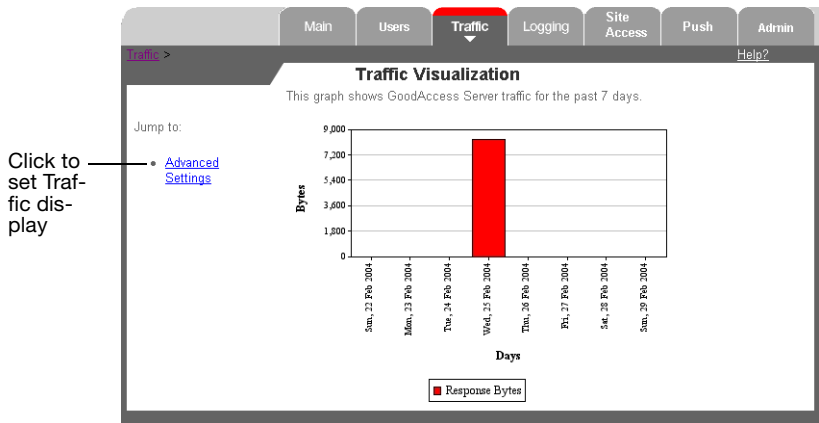
The user should be available for access.



## Traffic

The traffic page of the server console displays the amount of HTTP traffic sent between wireless handhelds and GoodAccess Server. This page is useful for comparing network traffic activity over hourly, daily, or weekly periods.

The following figure shows the default traffic display. This display shows response traffic sent to handhelds for the current day and the previous seven days.



**Important:** If the traffic page doesn't appear, make sure the traffic display is enabled in the *config.props* file. (See "Traffic Display" on page 152 for more information.)

### Setting Traffic Display Options

To set Traffic display options:

1. Click **Advanced Settings**.

The Traffic Visualization screen appears.

The screenshot shows the 'Traffic Visualization Advanced' configuration window. The 'Traffic' tab is active in the top menu. The window title is 'Traffic Visualization Advanced'. It features three main configuration areas: 'Chart Type' (Bar Chart selected), 'Data Frequency' (Days selected), and 'Plot Series' (Response checked). Date ranges are set from March 10 to March 12, 2004. A 'Create Chart' button is located at the bottom right of the configuration area.

2. Enter Traffic display options as desired. Options include:

#### *Traffic Display Options*

---

Item	Description
Chart Type	Displays traffic as either a bar chart, line chart, or tab-delimited text. You can paste the tab-delimited text into an Excel spreadsheet.
Data Frequency	Sets the time period that appears across the bottom of the display (along the X-axis). You can display data in hourly, daily, or weekly increments.
Plot Series	Controls whether HTTP responses, requests, or both are included in the display. In charts, requests and responses are differentiated by color.
Dates	Sets the start and end data for displayed data.

3. When you have finished entering options, click **Create Chart**.

## Logging

GoodAccess Server includes both production logs and diagnostics logs. With the exception of the Access log, production logs are enabled by default and cannot be disabled. Diagnostic logs are disabled by default and can be enabled if desired. The server also sends logging information to the NT Event log.

The logging page of the server console lists logs that have been enabled for the server and allows you to enable or disable logging for server components.

**Logging Management**

Jump to:

- Access
- Admin
- Content
- Default
- Push
- Stager
- Stdout
- Converter
- Diff
- Redirect
- ResponseSize

Items in bold are the current system settings. Not all logs may be enabled or disabled. Logs which may not be reset at runtime do not possess an Enable radio button.

**Log Directory**  
C:\Program Files\Good Technology\GoodAccess\Serverlogs

**Production Logs**

Log Name	Enabled	Yes	No	Filename
Access	Enabled	Yes <input checked="" type="radio"/>	No <input type="radio"/>	logs/access.log
Admin	Enabled	Yes <input checked="" type="radio"/>	No <input type="radio"/>	management.log

**Diagnostic Logs**

Log Name	Enabled	Yes	No	Filename
Converter	Enabled	Yes <input type="radio"/>	No <input checked="" type="radio"/>	no log
Diff	Enabled	Yes <input type="radio"/>	No <input checked="" type="radio"/>	no log
Redirect	Enabled	Yes <input type="radio"/>	No <input checked="" type="radio"/>	no log
ResponseSize	Enabled	Yes <input type="radio"/>	No <input checked="" type="radio"/>	no log

**Yes and No controls** only appear for logs that can be reset without restarting the server.

**Note:** In addition to enabling and disabling logs with the GoodAccess Server Console, you can use the *config.props* file to control logging behavior. For details, see “Server Configuration Parameters” on page 121.

### *Types of Logs*

The following table lists GoodAccess production and diagnostic logs.

#### *Types of Log Files*

---

##### **Production Logs**

Access	Logs information about HTTP requests and responses. Because this log tracks handheld access to your corporate network, Good Technology recommends checking this log periodically for security purposes.
Admin	Records initialization of the server console and errors.
Content	Logs content errors, such as incorrect HTML tags that prevent the browser from displaying a page.
Default	Contains start up, configuration, and shutdown information.
Push	This log stores information about the states of push messages and their deliverables.
Stderr	Logs serious error messages that would otherwise be sent to the standard error. This log should always be empty. If the server generates unexpected entries, save them and contact Customer Support at Good Technology.
Stdout	Primarily logs serious messages that would otherwise be sent to the standard output. Aside from initial startup entries, this log should always be empty. If the server generates any other entries, save them and contact Customer Support at Good Technology.

**Note:** With the exception of the Access log, production logs are always enabled.



*Types of Log Files*

---

**Diagnostic Logs**

Diagnostic logs are disabled by default. You can use the GoodAccess Server Console to enable these logs. Good Technology does not recommend using diagnostics logs in a production environment.

Converter	The server includes a set of content-conversion adapters for mapping text-based Web content to a compiled form that can be sent to handheld devices. The converter log stores messages related to content conversion.
Diff	Logs caching of new content and sending of differential data.
Redirect	Logs data when server encounters an HTTP redirection response from a Web server.
Urlsubst	Logs URL substitutions when the server replaces the path in a URL that follows the destination host name with a different path.

This table lists the most important logs created by the server. Additional logs may be enabled to support specific GoodAccess features. For example, if you use a proxy server for making HTTP requests, you can enable a proxy log to store proxy-related log messages. Also, you can enable an SSL log to track SSL activities (including trusted certificate loading). The SSL log can be run without greatly impacting server performance.

### *Log File Location*

All log files have a GoodAccess assigned name and are stored in server's log directory. The log directory location is set during installation and stored in the registry. By default, log files are located in:

*install\_dir/logs*

Individual log files are named using the following convention:

*install\_dir/logs/logname.log*

where *logname* identifies the type of log being maintained (for example, *responsesize.log*).

The most-recent logging information is always stored in files with the simple logging name (*logname.log*). When a log file is rolled over, the old file is closed and renamed with a time/date stamp. For example:

*install\_dir/logs/responsesize.log11-22-03.14-35-48*

where 11-22-03.14-35-48 is 2:35 PM and 48 seconds on November 22, 2003.

### *NT Event Log and Good Operations Log*

To maintain consistency with the GoodLink Management Server, GoodAccess Server is configured to send all log messages to the NT Event Viewer Application log. Also, the *good-ndc.log* file contains log events related to communication with the Good Operations Center. For more information on the *good-ndc.log* file, see the *GoodLink Administrator's Guide*.

### *Shared Log Messages*

In some cases, when communicating with the Good Operations Center, the GoodAccess Server and GoodLink Server share the same log messages. Even though the GoodAccess Server generates these messages, the term "GoodLink Server" may appear in the messages.

Shared messages include:

- logMsgXmlGwLoginFailed "GoodLink Server failed to login to GoodLink Operations Center as hostname <host\_name> Error code <error\_code>"
- logMsgXmlGwCannotConnect "GoodLink Server cannot connect to GoodLink Operations Center at URL <url\_name>. Confirm network access to Good Operations Center. Code <error\_code>"
- logMsgXmlGwHookupFailed "GoodLink Server failed to connect and authenticate with Good Operations Center at <location> with hostname <host\_name>. Reason code <error\_code>"
- logMsgXmlGwAckFailed "Send of acknowledgment of message received and processed failed from GoodLink Server to GoodLink network. Message: <message>. Code <error\_code>"
- logMsgXmlGwRemoveFailed "GoodLink Server failed to remove msg <message\_id> from Good Operations Center. Code <error\_code>"
- logMsgXmlGwAddQFailedQueueFull "GoodLink Server failed to send a message to connection <connection\_name> because the network queue is full."
- logMsgXmlGwCallbackThrew "GoodLink Server protocol engine callback to session manager threw an exception."
- logMsgXmlGwSessMgrRefusedMsg "GoodLink session manager refused message, id <message\_id>, from Good Operations Center. Code <error\_code>"
- logMsgXmlGwGetQFailed "GoodLink Server received an error fetching a message from the Good Operations Center. Code <error\_code>"
- logMsgXmlGwStatusReportFailed "GoodLink Server failed to report status to GoodLink Operations Center. Attempted to report status <status>. Got code <error\_code>"

### Site Access

The Site Access page allows you to:

- Limit internet access for users by disabling the “Go to Web Address” command on handhelds.
- Set the default home page for handhelds.
- Map the host name received in a URL to a different host name (Host Substitution).
- Map the path following the host name in a URL to a different path (URL Substitution).

URL substitution in combination with host substitution makes it possible to fully change the URL that a wireless handheld requests without reconfiguring the handheld. This can be useful, for example, if you want to change the server host.

The screenshot shows the 'Site Access' configuration page in the GoodAccess Administrator. The top navigation bar includes 'Main', 'Users', 'Traffic', 'Logging', and 'Site Access' (which is highlighted with a red tab). Below the navigation bar, the page title is 'Access to Sites' with the subtitle 'Substitutions Management'. On the left, there is a 'Jump to:' section with links: 'Walled Garden', 'Client Home Page', 'Host Substitutions', and 'URL Substitutions'. The main content area has two sections. The first section, 'Walled Garden', has a description: 'The walled garden allows you to limit access to sites using the GoodAccess Client. When walled garden is on, users will not be able to enter a URL from the "Go To Web Address" function.' Below this is a toggle switch for 'Enabled' with radio buttons for 'Yes' (selected) and 'No'. The second section, 'Client Home Page', has a description: 'Default Home Page for the GoodAccess Client.' Below this is a text input field containing 'http://www.google.com' and a 'set' button.

Substitute > Main Users Traffic Logging Site Access Help?

### Access to Sites

Substitutions Management.

Jump to:

- [Walled Garden](#)
- [Client Home Page](#)
- [Host Substitutions](#)
- [URL Substitutions](#)

**Walled Garden**

The walled garden allows you to limit access to sites using the GoodAccess Client. When walled garden is on, users will not be able to enter a URL from the "Go To Web Address" function.

Enabled Yes ☒ No ☐

**Client Home Page**

Default Home Page for the GoodAccess Client.

### *Configuring Internet Access for Users*

To configure internet access for users:

1. To limit access:
  - a. Locate the Walled Garden area of the page.
  - b. Select **Enabled > Yes**.  
When Walled Garden is enabled, the **Page > Go to Web Address** command on handhelds is disabled.
2. To specify the default home page on handhelds:
  - a. In the Client Home Page area, enter a home page address.
  - b. Click **Set**.

*Note:* This settings apply only to handhelds you set up *after* you configure Internet access on the server.

### *Host Substitutions*

In host substitution, server replaces the destination host name in a URL with a different host name. This page allows you to enable or disable host substitution, to specify host-name substitutions, and to delete previous host-name substitutions.

To set up host substitution:

1. In the Host Substitutions area of the page, select **Enabled > Yes**. Then, click **Set**.
2. Enter the name of the **From** host. This is the original destination hostname that appears in the URL.
3. Enter the name of the **To** host. This is the name of the new host to include in the URL.
4. Click **Add**.

The substitution you just added appears on the page. You can continue to add host substitutions as desired.

5. If you want to remove a substitution, click the **Remove** button next to the substitution.

The screenshot displays the configuration interface for Host and URL Substitutions. It is divided into two main sections: Host Substitutions and URL Substitutions. Each section has an 'Enabled' toggle (Yes/No) and a 'Current' list of substitutions. In the Host Substitutions section, the 'Enabled' toggle is set to 'No'. The 'Current Host Substitutions' list shows a substitution from 'localhost' to 'www.SomeOtherHost.com' with a 'Remove' button. In the URL Substitutions section, the 'Enabled' toggle is set to 'No'. The 'Current URL Substitutions' list shows a substitution from 'http://www.good.com/testhomedeck.wml' to 'http://www.good.com/homedeck.wml' with a 'Remove' button.

**Host Substitutions**  
Enabled Yes ☐ No ☒

**Current Host Substitutions**  
From  To  Add

From localhost  
To www.SomeOtherHost.com Remove

**URL Substitutions**  
Enabled Yes ☐ No ☒

**Current URL Substitutions**  
From  To  Add

From http://www.good.com/testhomedeck.wml  
To http://www.good.com/homedeck.wml Remove

### *URL Substitutions*

In URL substitution, the server replaces the path in a URL that follows the destination host name with a different path.

To set up URL substitution:

1. In the URL Substitutions area of the page, select **Enabled > Yes**. Then, click **Set**.
2. Enter the name of the **From** URL. This is the original destination URL.
3. Enter the name of the **To** URL. This is the name of the new URL.

#### 4. Click **Add**.

The substitution you just added appears on the page. You can continue to add URL substitutions as desired.

#### 5. If you want to remove a substitution, click the **Remove** button next to the substitution.

**Important:** URL substitution is applied before host-name substitution. Thus, to map the URL *http://localhost/intro.html* to *http://www.myvendor.com/demo/docdemo.html*, you need to keep *localhost* as the host name in URL substitution.

## Push

The Push page allows you to send and manage push messages to users enabled on the server. For more information, see “Using Push Technology” on page 67.

## Administration

The Administration page allows you manage login information for administrating the GoodAccess Server. You can use this page to change the default login and password for the server and allow additional users to perform server administration.

The screenshot shows the GoodAccess Administration console. At the top, there is a navigation bar with tabs: Main, Users, Traffic, Logging, Substitute, and Admin (which is highlighted with a red background and a dropdown arrow). To the right of the Admin tab is a 'Help?' link. Below the navigation bar, the page title is 'Administration >'. The main content area is titled 'Administration' and contains the following text: 'Administer the GoodAccess Management Console. Update console properties and configuration.' Below this text is a section titled 'Administration' with a dark background. This section contains five input fields: 'Current Username', 'Current Password', 'New Username', 'New Password', and 'Re-Enter New Password'. A 'Submit' button is located at the bottom right of this section.

### Stopping GoodAccess Server

Occasionally, you may need to stop GoodAccess Server to reset server properties, load new software, or troubleshoot server problems.

To stop the GoodAccess Server:

1. Open the Services panel.
2. Select **GoodAccess Server** and then click **Stop**.

When you manually stop GoodAccess Server, the Windows automatic recovery facility does not try to automatically restart the server.

### “Best Practices”

As with any mission-critical application, you will want to plan for optimal deployment, redundancy, backup, and disaster recovery for GoodAccess Server. This section describes or references procedures and rules for doing so.

### Backup

Be sure that you include the server in your backup procedures. In case of a disaster such as server corruption, you'll need to restore a copy of the server.

Backup procedures include:

- When you first install GoodAccess Server, the server makes a backup of the server configuration file and saves it as *config.props.bak*. You can use this file to restore the original server defaults if desired.
- Also, as you make changes to server configuration (using the GoodAccess Server Console interface or the *config.props* file), make sure you keep a current backup copy of the file.



- Make sure users backup critical data on their handhelds using the appropriate backup procedures. For example, on PPC handhelds, use a SD card for backup.
- Make sure you keep a current backup copy of any application properties files (*applicationname.props*) created for the server. For more information about these files, see “Developing and Extending Applications” on page 93.
- Make a backup copy of the list of users who have been enabled on the server.

**Note:** For additional backup and recovery procedures in a GoodLink environment, see the *GoodLink Administrator’s Guide*.

## Recovering from a Disaster

To start from scratch after a server disaster, follow this procedure:

1. Rebuild the operating system on the server host with the proper system requirements/prerequisites for the server. Use the same server name. It isn't necessary to use the same IP configuration.
2. Grant local administrator permissions to the *GoodAccessAdmin* account.
3. Log on to the server using the *GoodAccessAdmin* account and reinstall server software. Use the same license key, serial number and server name as for the previous server.
4. Start GoodAccess Server.
5. You should see all of the users you defined prior to the disaster. If not, you will need to re-enable user access to GoodAccess Server.
6. Send notifications to your GoodAccess handheld users. They may receive a message on their handheld asking them to reconnect to the server.



# 4 Preparing User Handhelds

---

In addition to installing the GoodAccess Server, you need to prepare user handhelds by installing GoodAccess software on the handhelds, and configuring user access from the server.

- You can install GoodAccess Server first and prepare user handhelds at a later date. This option is useful if you have custom applications you wish to test before deploying GoodAccess to a large number of users.
- GoodAccess handheld software is bundled with GoodLink 3.7 or later. You can install GoodAccess software on user handhelds, then install GoodAccess Server at a later date. This option is useful if you want to deploy GoodLink 3.7 to users before activating GoodAccess.

# Preparation Considerations

The following are some things to consider before you prepare user handhelds:

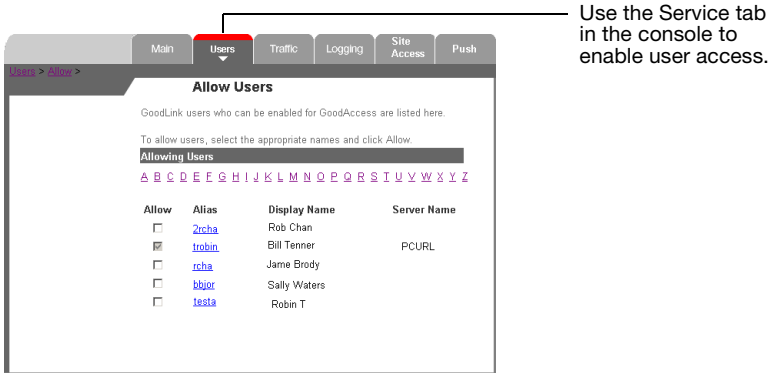
- A user must be provisioned on GoodLink before you can enable that user's account on GoodAccess Server.
- If the user's account is enabled for GoodAccess before GoodAccess software is installed on the handheld, the serial number and server name are automatically referenced when you start GoodAccess on the handheld. Otherwise, you (or the handheld user) will have to manually enter the serial number and server name.
- If you install GoodLink 3.7 (and GoodAccess 1.0) software on user handhelds before installing GoodAccess Server, you can send an email message to users with the correct serial number and server name once the server is installed.
- Because you install GoodAccess software as part of GoodLink 3.7 on user handhelds, you don't need to reinstall handheld software once GoodAccess Server is up and running.

# Installing GoodAccess Software on Handhelds

GoodAccess handheld software is bundled with GoodLink. The procedure for installing GoodAccess software on handhelds is the same as for GoodLink. You can use GoodLink Desktop or the GoodLink Management Console to install the software. See the *GoodLink Administrator's Guide* for details.

## Enabling User Accounts

In addition to installing GoodAccess Software on user handhelds, you must enable user accounts on GoodAccess Server. You use GoodAccess Server Console to enable user accounts. For details, refer to “Managing GoodAccess Server” on page 35.



## Setting Up GoodAccess on Handhelds

**Note:** If the user’s account was enabled on the GoodAccess Sever before GoodAccess software was installed on the handheld, you don’t need to set up GoodAccess; the application launches automatically.

To set up GoodAccess:

1. Open **GoodLink Preferences** on the handheld and choose **GoodAccess**.

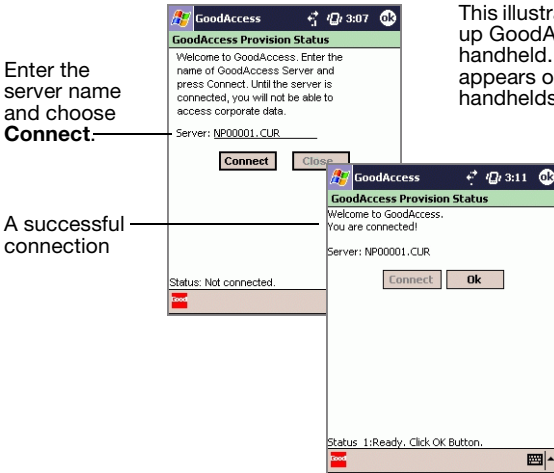
2. Select **Launch GoodAccess**.

A series of setup pages appears.

3. If required, enter the server connection information, then select **Connect**.

## Preparing User Handhelds

- After a connection is established, the **Close** button changes to **OK**. Select **OK**.



The GoodAccess main screen appears.

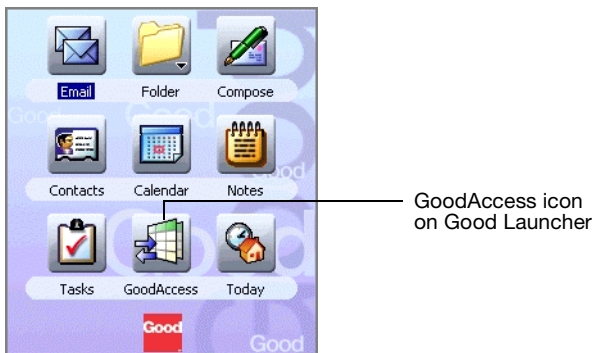
For more information on using GoodAccess on a handheld, see the “Using GoodAccess” chapter in the *GoodLink User’s Guide*.


## Starting GoodAccess on Handhelds

After GoodAccess is set up, you can use one of the following methods to start GoodAccess on handhelds.

**1. For Pocket PC handhelds:**

- Tap the **Good** icon  on the Today screen, then tap the **GoodAccess** icon on the Good Launcher, or
- Choose **GoodAccess** from the **Start** menu.



- 2. For Palm OS handhelds, tap the **Good** icon  on the Home screen. Then, tap the **GoodAccess** icon on the Good Launcher.**

## Uninstalling GoodAccess Software on Handhelds

GoodAccess handheld software is bundled with GoodLink. The procedure for uninstalling GoodAccess software on handhelds is the same as for GoodLink. You can use the Desktop installer or the GoodLink Management Console to uninstall the software. See the *GoodLink Administrator's Guide* for details.





# 5 Using Push Technology

---

## About Push

GoodAccess includes a push utility that enables system administrators to pro-actively broadcast pages, documents, and on-screen alerts to mobile users. Benefits of push messaging include:

- Push document can be traditional file attachments (such as Word, Excel, or PDF documents, HTML Web pages, or URL locations that contain intranet links, downloadable applications, and so on).
- Push documents can be downloaded and received as a background process, so a user's workflow is not interrupted.
- For important, immediate notification, administrators can send a push alert which automatically takes priority on a user handheld.
- Administrators can deliver push messages as needed or on a pre-defined scheduled basis.
- If the user receiving a push document or alert is out of coverage, the push is queued on the server until coverage is restored.
- Server queueing and handheld acknowledgement guarantee and confirm push delivery.

### Push Messages on the Handheld

Push messages and alerts appear in the incoming folder on the user's handheld. Users can leave the messages in that folder or save them to another folder.

*Note:* You cannot send bookmarks directly as pushed messages, but you can send URL locations that users can open and save as bookmarks.

When you send users an alert, a message appears at the bottom of the handheld screen. In addition, alerts trigger the notification method set for the handheld (for example, vibrate, alarm, or both.)

You can use push messages and alerts in combination with email. For example, you might have an automated bug tracking system that sends an email message to the appropriate service engineer when a serious bug is logged and a push message that includes a detailed description of the bug.

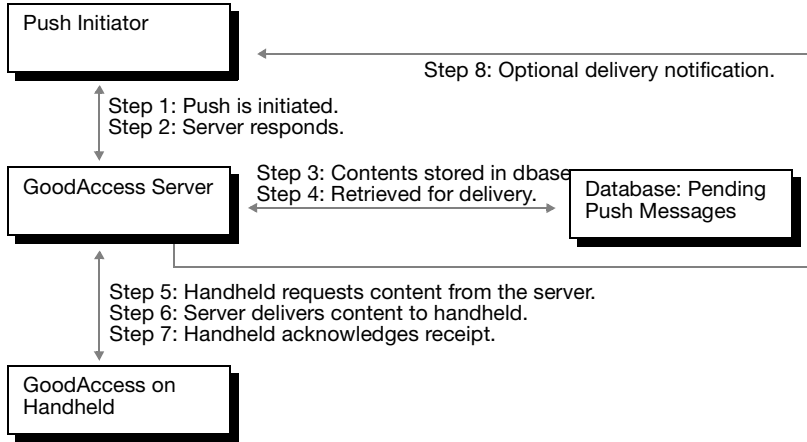
### Push Messaging Tools

GoodAccess includes both a command-line tool and a graphical interface for push messaging:

- The push command line tool is useful for scripting and batch processing. It enables you to send and manage push messages across multiple GoodAccess Servers. For more information on using the push command line tool, see "Using the Command Line Interface for Push Messaging" on page 75.
- The push graphical interface is available on the GoodAccess Server Console. It provides a simple, intuitive interface for sending and managing push messages on a single GoodAccess Server. For more information on using the push console interface, see "Process Overview" on page 69.

## Process Overview

The following is a brief overview of push processing:



1. The administrator uses a command-line utility, script, or the console interface to initiate a push to the user. GoodAccess uses PAP protocol to submit push messages via HTTP.
2. GoodAccess Server returns an HTTP (Ok or Not Ok) response and returns an ID for the pending push message.
3. The server queues the request in its local database of pending push messages.
4. When the push is ready for delivery (for example, at a pre-defined time), the server retrieves the message from the database and sends an indication to GoodAccess on the user handheld.
5. The handheld requests the content of the push message from the server.
6. The server delivers the content to the handheld.
7. The handheld acknowledges receipt of the push message.
8. Optionally, the server notifies the requestor about the status of the delivery.

# Using the Console Interface for Push Messaging

The GoodAccess Server Console provides a graphical user interface for sending and managing push messages. You can use the console to view the status of push messages, send new messages, generate push reports, and set push administration options.

**Important:** The GoodAccess Server Console sends and manages push messages for a specific GoodAccess Server (the server referenced by the console). Only users enabled on that server can receive messages sent using the console interface. For enterprise-wide messaging, use the push command line tool. See “Using the Command Line Interface for Push Messaging” on page 75.

## Viewing Push Messages in the Console

To view push messages:

1. Open the GoodAccess Server Console.

For more information, see “Opening the GoodAccess Server Console” on page 38.

2. Click **Push**.

A page appears with a list of push messages.

Click to view the Push interface.

Push messages sent from the server are listed here.

ID	Name	Type	URL	Submitted?	
1	Virus!	Immediately	http://patch/fixit	Yes	Edit Detail
2	SNote	Scheduled	http://snotes.march04.pdf	Yes	Edit Detail

3. Click **Details** to view more information about a message. Detailed information includes message delivery status, content location and so on.
4. To edit a message, click **Edit** to open the message. You can only edit unsubmitted messages.
5. To delete a message, click **Detail** to open the message, then click **Delete**.

## Sending Push Messages from the Console

To send push messages:

1. Open the GoodAccess Server Console.  
For more information, see “Opening the GoodAccess Server Console” on page 38.
2. Click **Push > Create New Push**.  
A page appears with a list of push settings.

Push > Create / Edit Push

### Create / Edit Push

Create / edit messages pushed to wireless handhelds.

**Jump to:**

- [View Existing Pushes](#)
- [Create New Push](#)
- [Generate Push Record](#)
- [Administer Push](#)

**Push Details**

ID: 2

Name: Service Bulletins

Title: Weekly Updates

Type:
   
☐ Immediate
   
☐ Scheduled
   
☐ Once
   
☒ Recurring

Start Date: 06/20/2004

Time: 13:40

Recurrence: ☒ Weekly ☐ Monthly ☐ On Content Change

### *Settings for Push Messages*

---

Item	Description
ID	A unique identifier for the push message. This number is created automatically.
Name	A user-defined name for the message. This name appears in message lists, and so on.
Title	Sets the document title displayed on the handheld.
Type	Set the type of message you want to create. You can create messages that are sent immediately or on a scheduled basis.
Start Date	Sets the time when delivery is first attempted.
Recurrence	Scheduled messages can be sent once or they can recur on a weekly or monthly basis.  You can also send scheduled messages that recur when URL content changes (for example, when a new Service Note is posted).
End Date	Sets the date when an undelivered message expires.
Alert	Indicates if the message should be sent as an alert. Alerts automatically take priority on the user's handheld.
Delivery Type	Sets where push content is stored on the handheld. Content sent to Doc Store appears in GoodAccess folders. Content sent to cache is displayed in the GoodAccess viewer.  <i>Note:</i> You cannot push contents to Bookmarks. Also, you cannot push Javascript or images to <i>docstore</i> . Pushed Javascript and images are stored in cache, regardless of the destination specified.
URL	The file or web resource that holds the push content.
Content Type	Indicates the type of document being pushed. For example, you can push excel, html, pdf, powerpoint, word, or wml documents.
Recipients	Email address of the users who will receive the push message. Only users enabled on this server are listed.

3. Enter settings as desired.
4. Click **Submit** to submit the push.

*Note:* You can click Save to save the message without submitting it. Saved messages can be reopened and submitted at a later date.

## Generating Push Reports from the Console

To generate a push report from the console.

1. Open the GoodAccess Server Console.  
For more information, see “Opening the GoodAccess Server Console” on page 38.
2. Click **Push > Generate Push Record**.  
A list of search criteria appears.

### *Search Criteria for Push Reports*

---

Item	Description
Push State	The message delivery state to include in the report (for example, delivered, cancelled, or expired). For a full list of delivery statuses, see “Delivery and Result Notification States” on page 90.
Recipient Address	Reports on messages sent to the email address specified.
Push ID	Reports on a specific push message based on the unique message ID.
Push Name	Reports on a specific push message based on message name.

3. Select search criteria as desired and click **Search**.  
A push report based on the search criteria you specified appears at the bottom of the page.

## Setting Push Administration Options

To set push administration options:

1. Open the GoodAccess Server Console.

For more information, see “Opening the GoodAccess Server Console” on page 38.

2. Click **Push > Administer Push**.

A list of options appears.

### *Push Administration Options*

---

Option	Description
Invoke Interval (in mins)	The interval (in minutes) between each recurring submit.
Status Interval (in mins)	Time interval in minutes that occurs between push status checks.
Clean Up Interval (in days)	Time interval (in days) between each scheduler clean up cycle.
Clean Up Windows (in days)	Clean up windows for push records and push address records in the push database (applies to non-pending pushes).
Start/Restart Delay (in mins)	The delay (in minutes) before the push GUI internal thread starts or restart.

3. Enter a new value for the option you want to change, then click **Change**.



## Using the Command Line Interface for Push Messaging

This section provides information on using the push command line interface. The push command line tool is useful for scripting and batch processing. It enables you to manage push messages across multiple GoodAccess Servers.

### Setting Up Push Messaging

The following is an overview of the steps you take to prepare for using push messaging on the command line. Step details will vary depending on how you plan to integrate push into your corporate environment.

To set up push messaging:

1. Determine which hosts on your network can be used to send messages. A host machine does not have to be running GoodAccess Server to send command line push messages.

To configure additional hosts for sending pushed messages, you'll need to update the *push.allowedhosts* property in the GoodAccess Server properties file (*install\_dir/config.props*) to include these hosts. For example:

```
push.allowedhosts host_1 host_2 host_3
```

For details, see "Configuring Hosts" on page 87.

2. Create a script (or some other application interface) to initiate push.

GoodAccess includes a command line utility to send push messages, but you can also include push commands in script (for example, a Perl script or *.bat* file).

Push commands include:

- Submit (Push)
- Cancel
- Status
- Listen
- Report

See below for a detailed description of push command syntax.

3. Determine the URL location for the file or web resource that contains the push content.
4. Preparing email addresses.

You can enter individual email addresses on the push command line. However, for multiple recipients, you may prefer to set up a side file or script with a list of email addresses.

5. Monitor push messages as desired.

You can use the Status command to check the status of individual messages or the Report command for detailed push reports.

## Invoking Push Commands

You can enter push commands directly on a DOS command line or include it in a script. To run from the command line:

1. Change to the GoodAccess Server installation directory.
2. Enter the following:

```
jre/bin/java com.good.push.tools.Command
```

Where **Command** is a specific push command with arguments.

For example:

```
jre/bin/java com.good.push.tools.Cancel -push-id=N56679
```

**Note:** For command options that require spaces within (or between) arguments, you need to put quotes at the beginning and end of the arguments. For example, `-address="jvc@office.comrmiller@oop.com"`

## Sending Push Messages

The Submit command sends a push message to the push server for delivery to handhelds. This message can be any content type and can be pushed to one or more handhelds (identified by email addresses).

To send a push message:

1. Create a URL location that contains the push content.
2. Use the following command to send a push message:

You can enter the command directly on a DOS command line or include it in a script.

Commands (with associated arguments and options) should be entered on one line (no carriage returns).

```
jre/bin/java com.good.push.tools.Submit -address="address..."
-url=url_syntax [options]
```

---

### Arguments

*-address="address..."*

### Description

Email address of the user receiving the push message. If you specify more than one email address, use a space to separate each address in the list. For example:

```
-address="jvc@office.com rmiller@oop.com"
```

At least one email address is required for the Submit command.

*-url =url\_syntax*

The file or web resource that holds the push content.

For a file use the **file:path** syntax. For example:

```
-url=file:saleslist.html
```

where *saleslist.html* is a file in the current directory.

For a web resource, use the following syntax:

```
http://host/<path>
```

Arguments	Description
Options include:	
<i>-alert</i>	Indicates the message will be delivered as an alert instead of a standard push message.
<i>-content-uri=value</i>	The document identifier on the handheld. If not given, the URL is used.
<i>-deliver-after=time</i>	Sets the time the delivery is first attempted. If no time is specified, delivery is started immediately. You can use any of the following values for time: <i>+hh:mm:ss</i> (a time relative to the current time). For example, +17:05:00. <i>mm/dd/yy hh:mm:ss</i> (absolute local time). For example, 12/22/03 07:15:00. <i>yyyy-mm-ddThh:mm:ssZ</i> (absolute UTC time). The T and Z are literal. For example, 2003-11-25T19:40:00Z
<i>-deliver-before=time</i>	Sets the time at which an undelivered message expires. If no time is specified, the server uses a default value.  Time values you can specify for the <i>-deliver-before</i> option are the same as for the <i>-deliver-after</i> option.
<i>-destination=[docstore   cache]...</i>	Sets where push content is stored on the handheld. For example: <i>cache</i> (device cache directory) <i>docstore</i> (offline document storage)  If no destination is specified, offline document storage is assumed.  <i>Note:</i> You cannot push contents to Bookmarks. Also, you cannot push Javascript or images to <i>docstore</i> . Pushed Javascript and images are stored in cache, regardless of the destination specified.
<i>-host=host_url</i>	The URL of the push host. If no host URL is specified, <i>http://localhost:5080</i> is assumed.
<i>-notify=notify_url</i>	A URL that receives result notification messages. If no URL is specified, notifications are not sent.

Arguments	Description
<i>-priority=</i> <i>"[low   medium   high]"</i>	Specifies a priority level for the message. For example, <i>-priority="high"</i> Higher priority messages are delivered first. If no priority is specified, the message has <i>medium</i> priority.
<i>-push-id=identifier</i>	Specifies the push identifier. If no identifier is specified, a unique push ID is generated automatically. This ID is used to log and track push messages.
<i>-title=doc_title</i>	Sets the document title displayed on handheld. If no title is specified, a GoodAccess-defined value is used.
<i>-type=doc_type</i>	Indicates the type of document being pushed. Values for <i>doc_type</i> include: <i>excel</i> <i>html</i> <i>pdf</i> <i>powerpoint</i> <i>word</i> <i>wml</i>  In addition to the specific types listed above, you can also specify other well-known types recognized by the handheld.  If no document type is specified, the handheld tries to infer the type where possible.

### Push Examples

The following are some examples of push messages. The commands (with associated arguments and options) are entered on one line (no carriage returns).

```
jre/bin/java com.good.push.tools.Submit  
-address="jjones@mycompany.com" -url=file:C:/service/note.htm
```

This example sends the *note.htm* file to *jjones@mycompany.com*.

```
jre/bin/java com.good.push.tools.Submit  
- address="jjones@mycompany.com marcher@mycompany.com  
rchen@lglegal.com" -url=file:local/alerts/help.wml -alert -title="Urgent!"
```

This example sends the *help.wml* file to three recipients (*jjones@mycompany.com marcher@mycompany.com rchen@lglegal.com*). The message is sent as an alert with an *Urgent!* title.

## Cancelling Push Messages

The Cancel command cancels a specified push message. You can cancel the message for all recipients or for one or more specified email addresses.

To cancel a push message:

1. Determine the push ID of the message you want to cancel.
2. Use the following command to cancel the message:

You can enter the command directly on a DOS command line or include it in a script.

```
jre/bin/java com.good.push.tools.Cancel -push-id=identifier [options]
```

---

Argument	Description
<code>-push-id=<b>identifier</b></code>	The unique identifier of the push message you want to cancel. This argument is required.
Options include:	
<code>-host=<b>host_url</b></code>	The URL of the push host sending the message. If no host URL is specified, <i>http://localhost:5080</i> is assumed.
<code>-address="<b>address...</b>"</code>	Email address of the user or users whose message should be cancelled. If you specify more than one email address, use a space to separate each address in the list. For example: <code>-address="jvc@office.com rmiller@oop.com"</code> If no address is specified, the message is cancelled for all recipients.

### *Cancel Examples*

The following are some examples of cancelling push messages. Commands (with associated arguments and options) should be entered on one line (no carriage returns).

This example cancels all deliveries of a pushed message:

```
jre/bin/java com.good.push.tools.Cancel -push-id=note.576
```

This example cancels a pushed message for two recipients.

```
jre/bin/java com.good.push.tools.Cancel -push-id=servicenote.87  
-address="jchen@company.com mhoft@sct.org"
```

### **Checking Push Status**

The Status command queries the push server for the status of a particular push message. You can check message status for all recipients or for one or more specified email addresses.

To check the status of a pushed message:

1. Determine the push ID of the message whose status you want to check.
2. Use the following command to check message status:

You can enter the command directly on a DOS command line or include it in a script.

```
jre/bin/java com.good.push.tools.Status -push-id=identifier [options]
```

---

Argument	Description
<code>-push-id=<i>identifier</i></code>	The unique identifier of the push message you want to check for status. This argument is required.
Options include:	
<code>-host=<i>host_url</i></code>	The URL of the push host that sent the message. If no host URL is specified, <i>http://localhost:5080</i> is assumed.
<code>-address="<i>address...</i>"</code>	Email address of the user or users whose message status should be checked. If you specify more than one email address, use a space to separate each address in the list. For example: <code>-address="jvc@office.com rmiller@oop.com"</code> If no address is specified, message status is checked for all recipients.

When you request push status, a message similar to the following appears in the command window:

```
Status Response from server http://localhost:5080
Push Id   : pubs.483
Address   : jvc@office.com
Code      : 1001
State     : pending
Desc      : Request has been accepted for processing
Event Time :
Address   : rmiller@oop.com
Code      : 1000
State     : delivered
Desc      : Request succeeded
Event Time : 2002-03-27T00:00:27Z
...
```



The Code field in the status contains a code value for the server's response to the push message as sent by the push initiator. The codes are specified in the WAP Push Access Protocol (PAP) specification. For example:

#### *Push Status Codes*

---

<b>Code</b>	<b>Description</b>
1000	The request succeeded.
1001	Push submission accepted for processing.
2000	Syntax error.
2001	Push submission refused.
2002	Address error.
2003	Address not found.
2007	Duplicate push ID.
3000	Server cannot process the push message because of an internal error.

#### *Status Examples*

This example gives the status of all deliverables in a push message ServiceNote.457:

```
jre/bin/java com.good.push.tools.Status -push-id=ServiceNote.457
```

### Generating Push Reports

In addition to getting push status for individual messages, you can also generate a push report. You can generate a report on:

- All push messages in the database.
- Push messages sent to specific email addresses.
- Push messages with specific content IDs.
- Push messages in specific states.
- Push messages with specific push IDs.
- Use the following command to generate a push report.

You can enter the command directly on a DOS command line or include it in a script.

```
jr/bin/java com.good.push.tools.Report [options]
```

---

Arguments	Description
Options you can use with Report include:	
<code>-address="address..."</code>	The report covers push messages for the email addresses specified. If you specify more than one email address, use a space to separate each address in the list. For example:  <code>-address="jvc@office.com rmiller@oop.com"</code>
<code>-content-uri="uri..."</code>	The report covers push messages that contain the specified content URIs. If you specify more than one URI, use a space to separate each.
<code>-delivery-state="state..."</code>	The report covers push messages that have the specified delivery states (for example, <i>expired</i> ). If you specify more than one delivery state, use a space to separate each.
<code>-push-id="identifier..."</code>	The report covers messages with the specified push IDs. If you specify more than one identifier, use a space to separate each.

**Arguments****-dateformat=*format*****Description**

Sets the time format used in the report. Time format options are *local* or *GMT* (Greenwich Mean Time). The default is *local*.

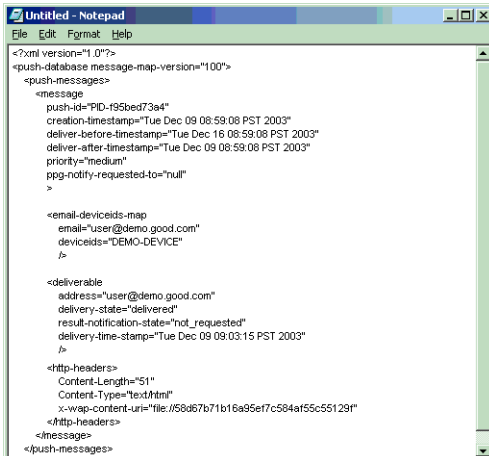
**-dbdir**

Specifies the push database directory. By default, this directory is *installdir/pushdb*.

*Note:* You must invoke the Report command from the GoodAccess Server installation directory or use a *-dbdir* option to specify the directory where the push database resides (by default this directory is *installdir/pushdb*).

**Report Output**

Push reports are supplied in XML format and include the push messages that meet the selection criteria specified by Report command options. The following is an example of a push report:



```

Untitled - Notepad
File Edit Format Help
<?xml version="1.0"?>
<push-database message-map-version="1.00">
  <push-messages>
    <message>
      push-id="PID-195bed73e4"
      creation-timestamp="Tue Dec 09 08:59:08 PST 2003"
      deliver-before-timestamp="Tue Dec 16 08:59:08 PST 2003"
      deliver-after-timestamp="Tue Dec 09 08:59:08 PST 2003"
      priority="medium"
      ppg-notify-requested-to="null"
    >

    <email-deviceids-map>
      email="user@demo.good.com"
      deviceids="DEMO-DEVICE"
    />

    <deliverable>
      address="user@demo.good.com"
      delivery-state="delivered"
      result-notification-state="not_requested"
      delivery-time-stamp="Tue Dec 09 09:03:15 PST 2003"
    />

    <http-headers>
      Content-Length="51"
      Content-Type="text/html"
      x-wap-content-uri="file:///58a67b71b16a95e7c584af55c55129f"
    </http-headers>
  </message>
</push-messages>

```

### Monitoring Notifications

You can use the Listen command to monitor and display result-notification messages sent by the push server. This command is useful for real-time tracking of changes in message status (for example, *delivered*, *expired*, and so on).

To monitor notifications:

1. When you use the Submit command to send a pushed message, make sure you include the *-notify* option as part of the command. For example, use *-notify=http://localhost:8081*.
2. Use the following command to monitor messages:

You can enter the command directly on a DOS command line or include it in a script.

```
jre/bin/java com.good.push.tools.Listen [-localport=port_number ]
```

---

Argument	Description
<i>-localport=port_number</i>	The port number to listen for result notification messages sent by the push server. The default port number is <i>8081</i> .

## Setting Push Properties

The following are properties you can set in the GoodAccess Server properties file (*install\_dir/config.props*) to control push behavior.

### Activation and Logging Properties

Use the following properties to activate push messaging:

---

Property Name	Default Value	Description
<code>push.enable</code>	<i>true</i>	If set to <i>true</i> , activates push messaging on GoodAccess Server.
<code>push.log.enable</code>	<i>true</i>	If set to <i>true</i> , enables logging of push messages.

### Configuring Hosts

Use the *push.allowedhosts* property in the *config.props* file controls which hosts will be allowed to perform push operations (using a Push Access Protocol (PAP) over HTTP interface).

The following are some example values:

---

Example	Description
<code>push.allowedhosts host-1 host-2</code>	Enables push messaging from <i>host-1</i> and <i>host-2</i>
<code>push.allowedhosts</code>	If no hosts are defined, push operations are denied on all hosts.
<code>push.allowedhosts all</code>	Allows push operations from all hosts.

If the *push.allowed hosts* property is not specified in the *config.props* file, push operations are allowed only from the machine running the server. This is equivalent to: *push.allowedhosts localhost*

### Properties for the Push Console Interface

The following additional properties appear in the *config.props*. These properties apply to the push console interface and are typically set from that interface.

---

Property	Description
push.ui.scheduler.invoke.interval	The interval (in minutes) between each recurring submit.
push.ui.scheduler.status.interval	Time interval in minutes that occurs between push status checks.
push.ui.scheduler.cleanup.interval	Time interval (in days) between each scheduler clean up cycle.
push.ui.scheduler.cleanup.windows	Clean up windows for push records and push address records in the push database (applies to non-pending pushes).
push.ui.scheduler.startup.delay	The delay (in minutes) before the push GUI internal thread starts or restart.

## The Push Database

GoodAccess Server maintains a database of push messages and push deliverables. A push deliverable corresponds to an instance of a push message that is to be sent to a specific handheld. For example, if the server sends a push message for three handheld devices, there will be three push deliverables corresponding to the push message.

Deliverables in the push database always have a delivery state. For example, when the server accepts a push message for delivery to one or more handhelds, each destination handheld has a corresponding deliverable and the state of each of these deliverable state is *pending*. When an instance of the push message associated with a deliverable is sent to a handheld and the handheld acknowledges receipt of the message, the deliverable's state is *delivered*. (For a list and descriptions of the states a deliverable can have in the database, see "Delivery and Result Notification States" on page 90.)

The push database is limited to push messages that are "current." Once a push message has been delivered to all target handhelds, it is deleted from the database. In addition, messages assigned a *deliver-before time* are deleted when they reach their deliver-before time, and messages over 30 days old are deleted.

GoodAccess Server provides two command-line utilities for accessing data in a push database:

- Report command

The Report Tool generates XML reports on the push messages in a push database. You can generate a report on all messages in the database, and you can use the tool to restrict the report to specific push messages, handhelds, content IDs, and delivery states (for example, *pending* or *delivered*). For information on the Report Tool, see "Generating Push Reports" on page 84.

- Status command

The Status command allows you to query the database for the current status of the deliverables in a specific push message. For more information, see “Checking Push Status” on page 81.

## Delivery and Result Notification States

The following table lists push delivery states:

*Push Delivery States*

Delivery State	Description
pending	Message from the push initiator has been accepted for delivery, but not yet sent.
delayed	Message accepted for delivery, but the message contains a <i>deliver-after</i> time, so delivery is delayed until that time.
enroute	Message sent over the air to the destination handheld, but an acknowledgment of delivery has not been received, yet.
delivered	The destination handheld has acknowledged receipt of the message.
replaced by newer deliverable	The message was not delivered because before delivery was requested, a new version of the same content was received from the push initiator and sent instead.
canceled	Message cancelled, either by the push initiator or through the Cancel command.
expired	Message could not be sent before the <i>deliver-before</i> time occurred.
rejected	The message was rejected because of a problem in the push message format.
specified push-id is not unique	The message was rejected because there is another message with the same push ID.
failed	The server was unable to deliver the message.



*Push Delivery States*


---

specified address is invalid	The server was unable to deliver the message because of an incorrect email address.
delivery attempts exceeded	The server sent the push message out five times without getting an acknowledgment from the handheld.

The following table lists push result notification states:

*Result Notification States*


---

Notification State	Description
notify delivered	The final delivery state of the message was successfully sent to the address specified in the push-message header's <i>ppg-notify-requested-to</i> attribute.
notify not requested	No result notification was sent because the push message did not contain a <i>ppg-notify-requested-to</i> address.
notify failed	The final delivery state could not be sent to the address specified in the push-message header's <i>ppg-notify-requested-to</i> attribute.



# 6 Developing and Extending Applications

---

After GoodAccess Server is installed, corporate developers can use GoodAccess to extend enterprise applications into the wireless arena. This chapter describes the GoodAccess application development process.

## About Transformations

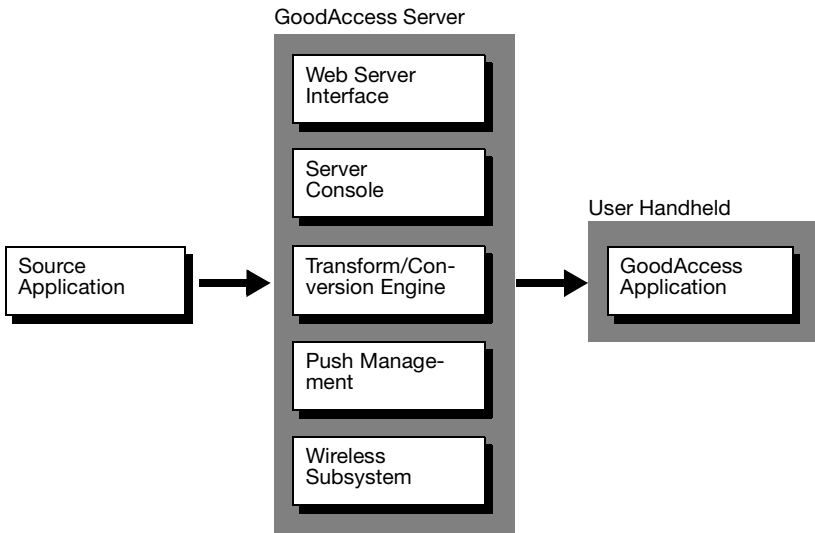
XSLT is an XML-based language used to create style sheets. GoodAccess Server includes a set of XSLT transformations you can use to integrate applications with GoodAccess. These transformations enable you to take new or existing corporate applications and optimize them for user handhelds. You can leverage your existing application development investment. You don't have to rewrite applications to work with GoodAccess.

Transformations are useful because many Web sites do not display well on a handheld's small screen. Transformations enable you to filter information before it reaches the handheld so users get the information they need in an attractive display. Transformations can be used on a specific page, a domain, or all pages in a site.

## Developing and Extending Applications

During transformation, GoodAccess technology parses through HTML to find specific items, remove items, or rewrite them. For example, GoodAccess can process a Web application to remove graphics or simplify table layouts. The XSLT style sheets defined for GoodAccess control the layout of the output documents and where to access data within the input document.

Transformations reside on the server, so all the heavy-duty processing is done by the server instead of the handheld. This enable more-efficient downloads and faster application processing on the handheld.



## Bibliography of Key References

To develop GoodAccess applications, you should be familiar with HTML, XML, and XSLT technologies. Good Technology also recommends (but does not require) you to be familiar with Java2 Platform Software and Java Server technology.

The following are some examples of books and Web sites you can use to learn more about the foundation technologies used with GoodAccess.

**Note:** Web locations are subject to change. If a URL listed is no longer active, try using a Web search engine to find the new location.

- For general information on HTML, XML, and XSLT, refer to the following books:
  - Mangano, Sal. *XSLT Cookbook*. First Edition, December 2002. O'Reilly & Associates, <http://www.oreilly.com>
  - Kay, Michael. *XSLT Programmer's Reference*. Second Edition, May 2001. Wrox Press, Inc., <http://www.wrox.com>
  - Tennison, Jeni. *Beginning XSLT*. First Edition, May 2001. Wrox Press, Inc., <http://www.wrox.com>
- For information on the Microsoft Pocket PC developer products, visit the Microsoft Library at <http://www.msdn.microsoft.com/library>
- For information on Apache open source software products, visit the Apache Web site at <http://www.apache.org>

## Overview of the Development Process

This section describes the overall development process for building and debugging application transformations. GoodAccess Server provides a set of default transformation tools you can use for your application or you can create your own custom transformations using XSLT technology.

- Identify a key application or Web site that mobile users will need to access when away from the office. Make a list of key information or program features users will need.
- Analyze the site you want to transform. Check the site to determine which pages should be transformed and which tools are best suited for the transformation. See “Picking Compatible Applications” on page 97 for details.
- Set GoodAccess Properties. Properties you set in the *config.props* file determine which URLs are transformed and which application properties files are used. See “Setting Up the GoodAccess Properties File” on page 97 for details.
- Create an application properties file. This file contains transformations specific to your application. You can include GoodAccess’ built-in transformations or point to a XSLT transformation file. See “Setting Up the Application Properties File” on page 99 for details.
- If necessary, design a custom XSLT transformation for your application. See “Creating Custom Transformations – An Example” on page 114 for details.
- Test and debug the application on a handheld. This chapter includes some tips and techniques for debugging common problems that can occur with transformations. See “Testing and Debugging Applications” on page 104 for details.

## Picking Compatible Applications

Transformations work on public Web sites or your own personal corporate application sites. One of the keys to preparing a successful transformation is to pick a compatible site.

Sites should include well-formed HTML. Not all public and private Web applications meet these requirements. For more information on well-formed HTML, refer to your HTML standards documentation.

Good Technology recommends starting simply then building complexity as your experience with transformations grows.

## Setting Up the GoodAccess Properties File

The *config.props* file is the first place the server looks for transformations. When you're creating or customizing a GoodAccess application, you need to edit the file (located in *install\_dir/config.props*) to include the following:

1. Define a selector property for your application.

The server uses a *selector* property to recognize which sites to transform.

Applications can be selected:

- Using regular expressions. For example:

```
app.gtswb.selectors requestURL=.*?[gG][tT][sS][wW][eE][bB].*
```

The server matches URLs by finding the best match. Be careful when you define a regular expression because the server could match similar expressions incorrectly.

- Using host matching. For example:

```
app.getit.selectors requestURL=http://w1thinkpad01/.*
```

- Using specific URL matching. For example:

```
app.demo.selectors requestURL=http://goodaccess/doc/addlabor.html
```

2. Set the location of the application properties file.

Once a site is selected by the server, the server references a *properties* setting for the site. This setting points to a specific properties file that defines transformations for the site.

The following are some examples of *properties* settings:

```
app.demo.properties app/gtsweb.props  
app.bugzilla.properties app/bugzilla.props
```

The *property* definition for a selector must match the selector exactly. So if you have a *gtsweb* application, you must have both the following properties defined:

```
app.gtsweb.selectors  
app.gtsweb.properties
```

3. Choose a reload option.

Optionally, you can have a *reload* property. When this property is set to *true*, you do not have to restart the server after making changes to the application properties file. For example:

```
app.demo.reload true
```

Good Technology recommend keeping this property set to *true*. Otherwise, you may experience unexpected application behavior.

4. When you've finished editing *config.props*, you should have completed set of property definitions for your application. For example:

```
app.myapp.selectors requestURL=.*  
app.myapp.properties app/myapp.props  
app.myapp.reload true
```

**Note:** For more information on setting properties in *config.props*, see "Server Configuration Parameters" on page 121.



## Setting Up the Application Properties File

After you edit the main Server properties file (*config.props*), you must create a more-specific properties file for your application (for example, *app/myapp.props*). Called an application properties file, this file contains transformation definitions for your application. Definitions can include the server's built-in transformations and/or pointers to custom XSLTs you have defined to selectively transform pages.

To set up an application properties file:

1. Create a plain text file.
2. Determine which built-in transformations you want to include in the file.
3. If necessary, create custom XSLT transformations and add pointers to these transformations. For more information, see "Creating Custom Transformations – An Example" on page 114.
4. Save the file. Make sure the filename and location matches the value you entered for the *app.name.properties* setting in *config.props*.

For an example of an applications properties file, see the generic application properties file located in the GoodAccess Server installation directory (*install\_dir/app/generic.props*).

## Using Built-In Transformations

GoodAccess Server includes the following built-in transformations you can use in your applications properties file. These transformations are automatically applied to *generic* applications (applications that don't reference any other properties file).

### *Built-In Transformations*

---

Transformation	Description
<i>CleanTables</i>	Removes empty rows, columns, and tables. Basically, removes empty table data cells (defined by <code>&lt;td&gt;</code> tags) from the document.
<i>CompressHtml</i>	Removes HTML content that holds redundant or unnecessary information.
<i>IndentTables</i> [( <i>style=ol</i> )]	<p>Replaces a table with a bulleted list. For example, a table with this structure:</p> <pre>a0 a1 a2 b0 b1 b2</pre> <p>Is changed to this structure:</p> <pre>a0 * a1 * a2 b0 * b1 * b2</pre> <p>If you specify the optional (<i>style=ol</i>) argument, the result includes a numbered list instead of a bulleted list (an HTML ordered list <code>&lt;ol&gt;</code>, instead of an unordered list <code>&lt;li&gt;</code>).</p>
<i>RemoveImgs</i>	Removes images ( <code>&lt;img&gt;</code> tags) that do not have hyperlinks. Also replaces images that are hyperlinked with the value of their <i>alt</i> attribute.
<i>TransformIfLarge</i> ( <i>size=number</i> )	Checks to see if the content is larger than the <b>number</b> of bytes specified. If so, further transformations are done.

*Built-In Transformations*

---

Transformation	Description
<i>TransformNever</i>	No further transformations are done. Because the <i>response.name.transforms</i> property must have at least one transformation specified, you can use <i>TransformNever</i> as a placeholder.
<i>Truncate (size=number [text=&lt;value&gt;])</i>	Limits the encoded content to no more than <i>number</i> bytes by removing HTML elements. If a text <i>value</i> is specified, this value is inserted into the content and appears on the handheld. If a text value is not specified, the default value, <i>Truncated by GoodAccess</i> , is used.  Example: <i>Truncate (size=22222 text="&lt;Truncated by Home Office&gt;")</i>
<i>UnpooledXslt (transform="filename")</i>	Transforms content like the <i>Xslt</i> transformation (described below). This transformation is intended for use during active development of the XSLT. It reads in the XSLT source every time it is used to transform a page.
<i>UnrollTables</i>	Moves the contents of table data cells (defined by <i>&lt;td&gt;</i> tags) outside the table. Inserts <i>&lt;br&gt;</i> tags as appropriate to maintain element spacing. For example, a table with this structure:  <pre>a0 a1 a2 b0 b1 b2</pre> <p>Is changed to this:</p> <pre>a0 a1 a2 b0 b1 b2</pre>

### *Built-In Transformations*

---

Transformation	Description
<i>UnrollTablesWithTables</i>	For tables within tables, the inner-most table layout is preserved, but all outer tables are unrolled.
<i>Xslt</i> ( <i>transform="filename"</i> )	Enables you to design and reference custom transformations. This command reads the XSLT style sheet specified by <i>filename</i> and transforms the content based on this style sheet. The <i>filename</i> value can be an absolute or relative path. (A relative path is relative to the directory that contains this application properties file.) Example: <i>Xslt (transform="bugzilla/bzMain.xsl")</i>

## Using Debugging Transformations

In addition to the transformations listed above, GoodAccess includes some debugging transformations you can use to help troubleshoot potential problems. See the *app/generic.props* file for details.

### *Unpooled Transformations*

GoodAccess Server also supports an *unpooledXSLT* transformation that refreshes automatically when changed.

---

<i>UnpooledXslt</i> ( <i>transform="filename"</i> )	This command reads the XSLT style sheet specified by <i>filename</i> and transforms the content based on this style sheet. This transformation refreshes automatically. The <i>filename</i> value can be an absolute or relative path. (A relative path is relative to the directory that contains this application properties file.) Example: <i>UnpooledXslt (transform="myapp/Main.xsl")</i>
--	---

## Using Custom Transformations

If desired, you can create custom transformations and add references to them in your applications properties file. For an example of how to create a custom transformation, see “Creating Custom Transformations – An Example” on page 114.

**Important:** If you create a separate properties files for your application, transformations (and other settings) in the *generic* application properties file no longer apply. If desired, you can copy (and modify) transformation settings from the *generic* application properties file to your application’s properties file.

## Testing and Debugging Applications

Before you deploy your application to users, make sure the HTML syntax of your application is correct. You can check the syntax by opening the application on a handheld then check the application interface and log files to see if errors occur.

This section describes logs, tips, and other tools you can use to help test and debug your applications.

### Log Support

GoodAccess Server includes a set of standard and diagnostic logs you can use to help test your applications:

#### *Types of Log Files*

---

Log Type	Description
----------	-------------

#### Production Logs

Access	Logs information about HTTP requests and responses.
Admin	Records initialization of the server console and errors.
Content	Logs content errors, such as incorrect HTML tags that prevent the browser from displaying a page.
Default	Contains start up, configuration, and shutdown information.
Push	Logs information about the states of push messages and their deliverables.
Stderr	Logs serious error messages that would otherwise be sent to the standard error.
Stdout	Primarily logs serious messages that would otherwise be sent to the standard output.

## Types of Log Files

---

Log Type	Description
----------	-------------

### Diagnostic Logs

Converter	The converter log stores messages related to content conversion.
Diff	Logs caching of new content and sending of differential data.
Redirect	Logs data when server encounters an HTTP redirection response from a Web server.
Urlsubst	Logs URL substitutions when the server replaces the path in a URL that follows the destination host name with a different path.

## Debugging Tips

The following are some tips on debugging transformations:

1. Refer to the stdout log to help debug a problem.  
This log contains useful information about how the server is matching your transformations.
  - a. Check the GoodAccess Server Console or the *config.props* file to make sure the stdout log is enabled.
  - b. Examine the log entries to see how transformations are being applied. (See example below.)
2. Include a transformation identification stamp in the application file.
3. In custom XSLT transformations, use lowercase only to match element names and attribute values. For example:

```
<a href="http://www.name.com">Name</a>
```

instead of:

```
<A HREF="http://www.name.com">Name</A>
```

### Using Transformation Debugging Tools

Because of wireless connectivity and provisioning requirements, developing and testing content transformations can be difficult using an actual handheld. Good provides the following tools to help you test and debug transformations:

- A command-line tool (*HttpTransformTool*) for doing scripted tests.
- An HTTP proxy tool (*HttpTransformProxy*) for interposing between Internet Explorer (or other browsers) and the server.

Both tools can be run from a local, networked computer that has access to GoodAccess Server. They show the following:

- Request made to the origin server
- Response returned from the origin server
- Tidied response
- Tidied and transformed response
- Debug output for the transformation selection

To use these tools, you must have the HTTP development port enabled on your GoodAccess server and a Web browser configured for proxying through the tool. Instructions are provided below.

**Important:** Good Technology provides these transformation tools for your convenience only. Further testing and debugging is required for creating transformations that work in a production environment.

#### *Downloading the Transformation Tools*

Both tools are packaged in a large ZIP file that includes an entire Java Runtime Environment (JRE). This packaging makes it easier to run the tool remotely. For example, you can run the tool on your local computer and talk to a GoodAccess Server elsewhere on the network.



To download the tools:

1. The tools are available at a GoodAccess FTP site. You can FTP to this site from a local machine.
2. Create an FTP connection in binary format to:  
     Server: *ftp.good.com*  
     Username: *goodaccess*  
     Password: *G00D4access* (use zeros for the Os)
3. Change to the *GoodAccessServer* directory.
4. Use the FTP connection to download the ZIP file (*gatools\_4.exe*) to your local machine.

### *Setting Up GoodAccess Server*

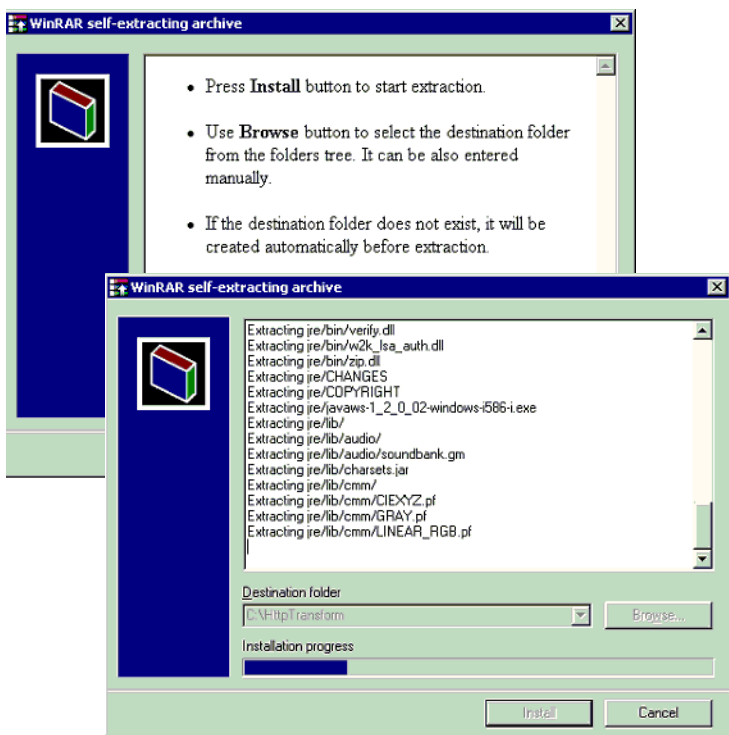
To set up GoodAccess Server:

1. Log on to the host machine that runs GoodAccess Server.
2. Change to the GoodAccess Server installation directory. For example:  
     *C:\Program Files\Good Technology\GoodAccess Server*
3. Add the following properties to the *config.props* file:  
     *httpdev.enable true*  
     *httpdev.localport 8090*  
     These settings enable an HTTP development port on the server.
4. Restart GoodAccess Server.

### *Installing the Transformation Debugging Tools*

To install the tools:

1. On the local machine, navigate to the directory where you downloaded *gatools\_4.exe*.
2. Double click the file icon to open the file and extract its contents. You'll see the following screens. Follow the instructions as directed.



The files are extracted and the tools are installed in the directory you specified.

### *Using the HttpTransformTool*

The HttpTransformTool is a command line tool you can use for scripted tests.

To use the tool:

1. Open a command window on the local machine.
2. Change to the directory where you installed the transformation tools.
3. Type the following command:

```
jre/bin/java HttpTransformTool
```

The following text appears. It describes how to use the tool.

*Usage:*

```
HttpTransformTool -url=<url> [options]
```

*Description:*

*Make an HTTP "get" request through the GoodAccess Server, returning information about the request and its transformation*

```
-url=<url>
```

*The web resource to "get". The <url> has the form*

*<http://host/<path>>*

*Options:*

```
-host=<url>
```

*The URL of the development port on the GoodAccess Server.*

*If not given, the value "<http://localhost:8090>" is used.*

4. Follow the on screen instructions to use the tool.

When you enter the HttpTransformTool command and specify a URL it should connect to the GoodAccess Server and return a multi-part MIME document as a result.

### *Using the HttpTransformProxy*

The HttpTransformProxy tool enables you to use a Web browser on your local machine to view transformed XHTML content produced by GoodAccess Server.

To use the tool:

1. From the local machine, change to the directory you used to install the tools:
2. Type the following command:

```
jre/bin/java HttpTransformProxy
```

The following text appears describing how to use the tool:

*Usage:*

```
HttpTransformProxy -localport=<port>[options]
```

*Description:*

*Set up a proxy server that forwards requests made by a web browser to the GoodAccess development port, causing the request to look like a GoodAccess handheld request. The response returned to the web browser is by default) the transformed XHTML produced by the GoodAccess server*

*Required arguments:*

```
-localport=<port>
```

*<port> is the numeric value for this proxy port.*

*Options:*

```
-host=<url>
```

*The URL of the development port on the GoodAccess Server.*

*If not given, the value "http://localhost:8090" is used.*

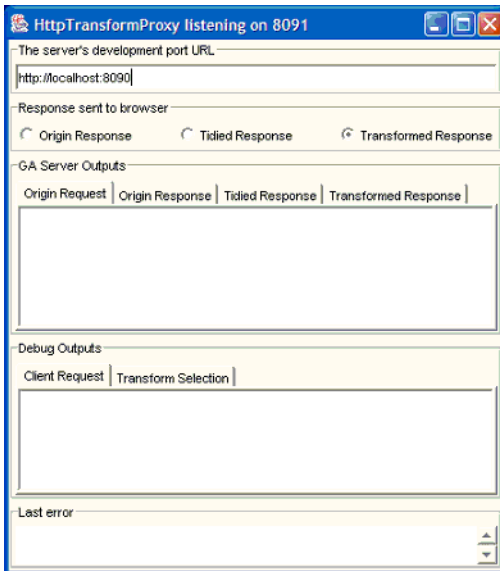
```
-response="{original | tidied | transformed}"
```

*Selects the type of response returned to the requestor. By default the transformed XHTML is returned*

3. To start the tool on port 8091, type the following:

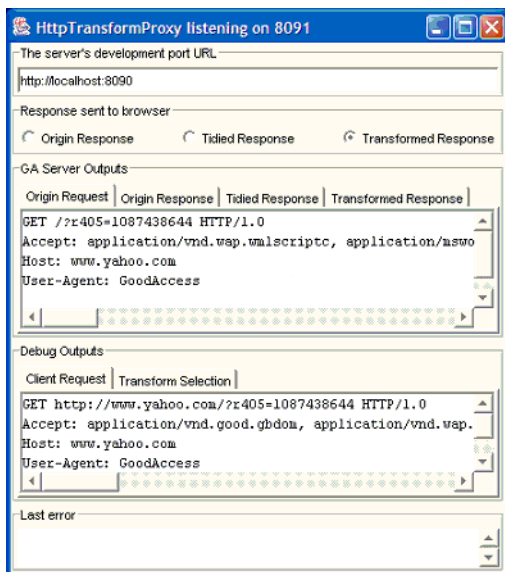
```
jre/bin/java HttpTransformProxy -localport=8091
```

The following interface appears:



4. Configure your web browser to proxy to port 8091 on *localhost*.  
For Internet Explorer:
  - Open **Tools > Internet Options > Connections > LAN Settings**.
  - Select **Use a proxy server**. Enter **localhost** for the address and **8091** for the port.
  - Save your settings.
5. To see it working, open your Web browser and navigate to a URL. For example, type [www.yahoo.com](http://www.yahoo.com).

You should see a response in the browser and several results in the tool interface. For example, the Original Request pane will show something like the following:



This is the actual request sent to the web server by GoodAccess Server. You can resize the tool as needed.

## Deploying GoodAccess Applications

When you have finished test and debugging your GoodAccess application, you're ready to deploy the application to user handhelds.

To deploy a GoodAccess application:

1. Post the application on an HTTP (or HTTPS) site.
2. Email the URL location to mobile users.
3. The mobile user clicks the URL in the email message to access the application.

## Creating Custom Transformations – An Example

The following example shows how to create a custom transformation for the Flight Status page on *www.yahoo.com*. You can use this as a model for creating your own custom transformations.

**Important:** Information provided in the chapter is for instructional purposes only. Public Web sites are subject to change, and if so, some portions of this example may no longer apply. All Yahoo!® trademarks, logos, service marks, trade dress, slogans, copyrighted designs, or other brand features can be used only as explicitly licensed by Yahoo!.

The main tasks required to create a custom transformation are:

- Analyze the site.
- Update the application properties file to apply a custom XSLT transformation.

**Important:** Do not confuse the application properties file (for example, *install\_dir/app/flightstatus.props*) with the main server properties file (*install\_dir/config.props*).

- Create a custom style sheet for the site.

Each of these tasks are described in detail below.

### Analyzing the Site

The first step in creating a custom transformation is to look at the site and determine the type of transformations you want to perform.

1. Examine the application interface for important information content.



You'll notice on the flight status page that most of the relevant information is located on the right, near the center of the page.

Relevant information

Reproduced with permission of Yahoo! Inc. \* 2004 by Yahoo! Inc. YAHOO! and the YAHOO! logo are trademarks of Yahoo! Inc.

### 2. Examine the application source code for key fields and forms.

There are two HTML form definitions that contain the relevant fields in the application.

Corresponds to first form definition

Corresponds to second form definition

Reproduced with permission of Yahoo! Inc. \* 2004 by Yahoo! Inc. YAHOO! and the YAHOO! logo are trademarks of Yahoo! Inc.

- a. This is the form that holds the top part of the application for users who know the airline and flight number for the status check:

```
<form name="status_resv_lhc_us" action="http://  
edit.travel.yahoo.com/config/ytravel_checkflight" method=get >
```

- b. This is the form that holds the bottom part of the application for users who need to look up flight information before getting status:

```
<form action="http://edit.travel.yahoo.com/config/  
ytravel_checkflight" method="GET" name="RESERVE" >
```

**Tip:** To help search through HTML code for relevant fields and forms, try searching the text for distinguishable keywords such as “form,” “method” or “get.”

## Updating the Applications Properties File

Before you define a custom transformation, make sure the applications properties file (*app/appname.properties*) includes a pointer to the transformation.

1. Open the application properties file. For example:

```
install_dir/app/flightstatus.props
```

2. Add a selector for the page. For example:

```
response.yahooflightmain.selectors  
requestURL=http://edit.travel.yahoo.com/config/ytravel_checkflight
```

3. Add a custom transformation:

```
response.yahooflightmain.transforms  
XSLT (transform="flightstatus/custom.xsl")
```

## Creating a Custom Style Sheet

This section describes the custom XSLT style sheet you need to parse out the two forms described above.

1. Begin the style sheet with an XML header statement. This statement is required for all XML files:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

2. Continue with a style sheet definition statement:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
```

3. Add statements for title, body and template matching criteria:

```
<xsl:template match="/"> _____ This line specifies which
<html>                               nodes to process.
<head>
<title>Flight Status</title>
</head>
<body>
<xsl:apply-templates select="*" /> _____ This line is important. It says
</body>                               to apply every applicable
</html>                               template in the style sheet.
```

4. Define a template that parses the *status\_resv\_lhc\_us* form.

```
<xsl:template match="//form[@name='status_resv_lhc_us']">
<form> _____ This line searches every
<xsl:copy-of select="*|@" />           node for a form with this
</form>                               name attribute.
</xsl:template> _____ This line tells the parser to
                                   print all information in the
                                   form.
```

In the example above, the `[@name='status_resv_lhc_us']` statement basically means, "search every node for a form with the *status\_resv\_lhc\_us* name attribute.

5. Define a template that parses the RESERVE form.

```
<xsl:template match="//form[@name='RESERVE']">
  <form>
    <xsl:copy-of select="*" @*"/>
  </form>
</xsl:template>
```

Again, this line tells the parser to print out all information in the form.

6. Define a final template that parses out the remaining page contents.

```
<xsl:template match="*" @* | text()">
  <xsl:apply-templates select="*" @* | text()"/>
</xsl:template>
```

The first line matches for contents that hasn't been matched before. The second line recurses. The result is removal of elements that match.

7. Add a closing statement for the style sheet.

```
</xsl:stylesheet>
```

8. Save your custom transformation and copy the file to the location you set in the main application properties file. For example:

```
install_dir/app/flightstatus/custom.xsl
```

## Processing Result

After you test and debug the application, the following is an example of the how the flight simulator page should appear on a user handheld.

**Locate a specific flight**

Choose the airline:

Flight number:

---

If you do not know the airline and flight number, use the fields below:

Leaving from:

Going to:

Date of

Simplified interface on handheld

Reproduced with permission of Yahoo! Inc. \* 2004 by Yahoo! Inc. YAHOO! and the YAHOO! logo are trademarks of Yahoo! Inc.

For more information on deploying and debugging applications, see “Testing and Debugging Applications” on page 104.

## Customizing Tips

The following are some things to remember when you are creating custom transformations:

- Telling the server which urls to match and transforming pages are two separate processes.
- The `<xsl:apply-templates select="*" />` and `<xsl:template>` statements included in a custom transformation are standard XSLT functions.
- All content is added by default. Use the following code to remove everything you don't want:

```
<xsl:template match="*" | @* | text()">
<xsl:apply-templates select="*" | @* | text()" />
</xsl:template>
```

- Keying in on specific attributes is the easiest way to transform:

```
<xsl:template match="//form[@name='RESERVE']">
  <form>
    <xsl:copy-of select="*|@*" />
  </form>
</xsl:template>
```

- Use lowercase only to match element names and attribute values. For example:

```
<a href="http://www.name.com">Name</a>
```

instead of:

```
<A HREF="http://www.name.com">Name</A>
```

- Do not strip the content between `<head>....</head>`. This often includes Javascript function definitions or globally-executed Javascript code.
- Do not remove form elements in a form, or be careful about doing so when the Javascript on a page contains references to offsets within that form. For example, if a script refers to an element like the following:

```
window.document.myform.elements[163].value = "foo";
```

and the element has been stripped by a transformation, the page may not work correctly.

- Be careful about removing forms from pages. Sometimes forms are referred to by their offsets. For example:

```
var theForm = document.forms[2];
```

- For more details on creating custom transformations or integrating your applications on GoodAccess, contact Good Technology's Professional Services department.

# 7 Server Configuration Parameters

---

Server configuration properties can be set from the GoodAccess Server Console or in configuration files. Commonly-used properties are included in the server console and a more extensive set of properties are included in configuration (also called “properties”) files.

This chapter describes setting properties in the server properties files. For information on setting properties in the GoodAccess Server Console, see “Managing GoodAccess Server” on page 35.

**Important:** If you change configuration settings directly in a properties file, make sure you use a plain text editor (such as Microsoft Notepad) to edit the text. Also, make sure you save a backup copy of the file before making changes.

# About Properties Files

GoodAccess Server contains the following property files:

- Default property file (*config.props*). Contains the most-common Server properties and defaults.
- Application properties files (*applicationname.props*). These are properties files you can create for your applications. GoodAccess includes a generic properties file (*generic.props*) that applies to all applications that don't have an application-specific properties file.
- Advanced property file (*advanced.txt*). Contains advanced properties for fine-tuning your server installation. Do not use these properties unless directed by Good Technical Support.
- When you first install GoodAccess Server, the server makes a backup of the server configuration file and saves it as *config.props.bak*. You can use this file to restore the original server defaults if desired.

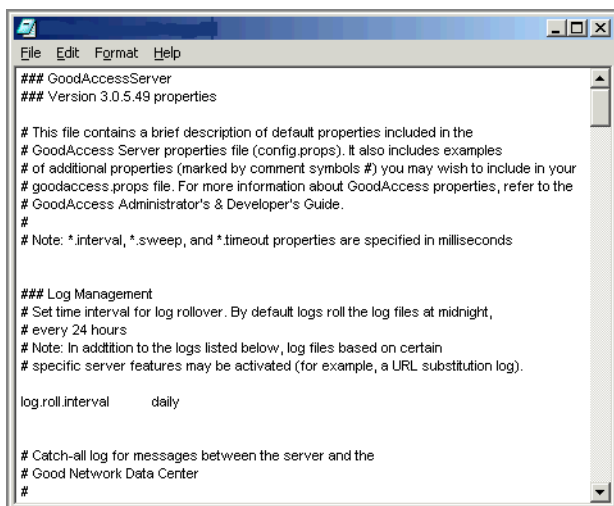
**Important:** Only the default properties file (*config.props*) and associated application properties files (*applicationname.props*) are actively referenced by GoodAccess Server during startup. To use a supplemental property described in *advanced.txt*, remove the comment marking (#) in front of the property and copy it to *config.props*.



## The Default Properties File

Most configuration options described in this chapter are located in the *config.props* file. This file is located in:

*install\_dir/config.props*



## Server Configuration Parameters

The table below describes major sections in the file and where to go for further information about properties included in these sections:

### *Sections of the GoodAccess Properties File (config.props)*

---

Section	Description	For More Information
Client Configuration	Sets the default GoodAccess home page and internet access permissions for handhelds.	"Client Configuration" on page 127.
Content Conversion	Enables the server to optimize content sent to handhelds by compiling and compressing content.	"Content Conversion and Transformation" on page 128.
Application Support	Determines how applications and URLs are selected for content transformation. Controls properties files used by applications.	"Application Support" on page 165.
Host Substitution	Maps a specified hostname in a URL to a different hostname. You can also use the GoodAccess Server Console to set host substitution.	"Host Substitution" on page 130.
URL Substitution	Maps a specified URL path to a different path. You can also use the GoodAccess Server Console to set URL substitution.	"URL Substitution" on page 132.
Redirect	Enables GoodAccess Server (rather than handhelds) to handle URL redirection responses from the Web server.	"Redirected HTTP Requests" on page 134.
Header Management	Enables the server to modify HTTP header requests before forwarding the requests to their destination hosts.	"Header Management" on page 137.
Differential Data	Enables the server to incrementally update documents stored on handhelds.	"Differential Data" on page 144.

### *Sections of the GoodAccess Properties File (config.props)*

---

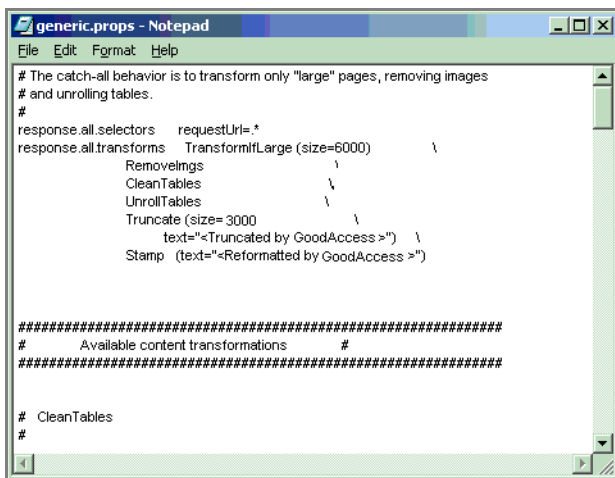
<b>Section</b>	<b>Description</b>	<b>For More Information</b>
Response Size	Enables the server to limit the amount of HTTP response content sent to handhelds.	"Response Size" on page 148.
Push Support	Enables the server to send push messages to handhelds. Also controls which hosts are allowed to initiate a push command.	"Setting Push Properties" on page 87.
Server Console	Contains properties that control the GoodAccess Server Console. This includes properties to enable the console, setting the console port, and console login.	"Server Console" on page 150.
Traffic Display	Enables the GoodAccess Server Console to display the amount of data traffic sent to and from handhelds.	"Traffic Display" on page 152.
Log Management	Contains properties that control server logging. This includes enabling and disabling different types of logs.	"Log Management" on page 153.
SSL Support	Enables Secure Socket Layer (SSL) communications (https) between the server and secure Web sites.	"SSL Support" on page 157.

### Application Properties Files

In addition to the *config.props* file, applications you develop for GoodAccess can have their own properties file to control data transformations, header replacement, and so on. Application properties files are located in:

**install\_dir/app**

This directory includes a *generic.props* file that applies to all applications that don't have an application-specific property file. You can use this file as a template for creating custom properties files for your own applications.



## Client Configuration

These properties enable you to:

- Limit internet access for users by disabling the **Go to Web Address** command on handhelds.
- Set the default home page for handhelds.

Client configuration properties include:

---

Property Name	Default	Description
client.config. GA_HomePage	<i>http:// www.good.com</i>	Sets the home page URL for GoodAccess handhelds. You can use an external URL (if internet access is enabled) or an internal URL.
client.config. GA_WalledGarden	<i>false</i>	Enables internet access restrictions. If set to <i>true</i> , the <b>Go to Web Address</b> command on the handheld is disabled.

# Content Conversion and Transformation

GoodAccess Server uses content conversion and transformation procedures to reformat Web content. These procedures enable the server to prepare information for optimal display on a wireless handheld.

Content conversion is the process of changing Web content into a compact format for delivery to handhelds. Content transformation is the process for reformatting, truncating, and otherwise scaling down large HTML pages to make them suitable for viewing on the small screen of handhelds.

## Content Conversion Adapters

GoodAccess Server includes a set of content-conversion adapters for mapping text-based Web content to a compiled form that can be sent to handheld devices. Compiled content is more compact and far more efficient for wireless transmission than text-based content.

There are content-conversion adapters for the following types of text-based content:

- HTML
- Wireless Markup Language (WML)
- Plain text
- WMLScript (a scripting language for WML)

Content conversion is always enabled, GoodAccess Server automatically detects the content type and compiles the content using the appropriate conversion adapter.

## Content Transformation

Content transformation properties are stored in separate application-specific configuration files (not in *config.props*). Conversion and transformation properties differ for different applications and different URLs. For more information on content transformation properties, see “Developer’s Reference” on page 165.

The following property enables the use of converters:

---

Property Name	Default	Description
converter.enable	<i>true</i>	If set to <i>true</i> (the default), enables content conversion. <i>Note:</i> The converter must to be enabled at all times.

# Host Substitution

Host substitution makes it possible to change the Web site that a wireless handheld requests without having to reconfigure the handheld. This can be very useful when changing server hosts.

With host substitution, you can set GoodAccess Server to map a specified host name in a URL to a different (substitute) host name that you also specify.

For example, if a wireless handheld is configured to initially send all of its HTTP requests to *localhost*, where *localhost* is GoodAccess Server, the server can be configured to send the request to a different host, such as *www.gadocs.com*.

```
hostsust.enable true  
hostsust.sub localhost www.gadocs.com
```

The first line in the example enables host substitution; the second line specifies that requests to *localhost* are to be sent to *www.gadocs.com*.

Any time the handheld's requests need to be redirected to a different host, the *hostsust.sub* property can be changed to specify the new destination.

You can specify multiple host-name substitutions by entering pairs of URLs as a space-separated list. In the following example, *localhost* is mapped to *www.gadocs.com* and *www.abcdef.com* is mapped to *www.zyxwvu.com*:

```
hostsust.sub localhost www.gadocs.com  
www.abcdef.com www.zyxwvu.com
```



Host substitution properties include:

Property Name	Default	Description
hostsubst.enable	<i>false</i>	If set to <i>true</i> , enables host substitution.
hostsubst.sub	<i>localhost</i> <i>www.SomeOtherHost.com</i>	One or more host substitutions, specified as: <i>host new_host host new_host...</i> For example, to forward requests sent to <i>www.gate.com</i> to <i>www.goodaccess.com</i> , enter: <i>hostsubst.sub www.gate.com</i> <i>www.goodaccess.com</i>
hostsubst.log. enable	<i>false</i>	If set to <i>true</i> , enables logging of host substitution events.

# URL Substitution

URL substitution (in combination with host substitution) makes it possible to fully change the URL that a wireless handheld requests without reconfiguring the handheld.

If a wireless handheld is configured to initially send all of its HTTP requests to *localhost/home*, where *localhost* is GoodAccess Server, the server can be configured to forward the request to a new location – *differenthost/differenthome*.

For example, with URL substitution, you can set GoodAccess Server to map the path in a URL that follows the host name to a different (substitute) path that you also specify.

```
urlsubst.enable true
urlsubst.sub localhost/intro.html localhost/demo/docdemo.html
```

The first line in the example enables URL substitution. The second line specifies that requests to *localhost/intro.html* are to be redirected to *localhost/demo/docdemo.html*.

**Important:** URL substitution is applied before host-name substitution. Therefore, to map the URL *http://localhost/intro.html* to *http://www.rcdocs.com/demo/docdemo.html*, you need to keep *localhost* as the host name in URL substitution, as shown in the example above, and configure host-name substitution to map *localhost* to *www.gadocs.com*, as described in “Host Substitution” on page 130.

You can specify multiple path substitutions by entering pairs of URLs as a space-separated list. For example:

```
urlsubst.sub localhost/intro.html localhost/demo/docdemo.html www.
abcdef.com/pubs.html www.abcdef.com/engineering/techdocs.html
```

In this example, *localhost/intro.html* is mapped to *localhost/demo/docdemo.html* and *www.abcdef.com/pubs.html* is mapped to *www.abcdef.com/engineering/techdocs.html*.

Properties for URL substitution include:

Property Name	Default	Description
urlsubst.enable	<i>false</i>	If set to <i>true</i> , enables URL substitution.
urlsubst.sub	<i>http:// www.goodaccess.com/ testhomedeck.wml</i>  <i>http:// www.goodaccess.com/ homedeck.wml</i>	The URL substitution that you want to be carried out, specified as:  <i>hostname/path hostname/ substitute_path</i> For example, to redirect requests sent to <i>www.gatenga.com/home/news</i> to <i>www.gatega.com/new_home/info</i> , you would enter:  <i>www.gatenga.com/home/news www.gatega.com/new_home/info</i>  <b>Important:</b> the host name must be the same in both URLs; only the path following the name can be replaced. To change the host name in a URL, you need to use host substitution.
urlsubst.log.enable	<i>false</i>	If set to <i>true</i> , enables logging of URL substitution events.

# Redirected HTTP Requests

In some cases, a Web server is configured to redirect requests for a particular URL to a different location. For example, if a company's home page is moved to a new location, requests for the old URL may be directed to the new URL.

To save valuable, over-the-air time, you can configure GoodAccess Server to handle redirection responses.

When a Web server makes a redirection response, it sends a message with an HTTP response code of 301 or 302 (the location of the requested Web page has changed, either permanently or temporarily) and the URL of the new location. The recipient of the redirection response can then send an HTTP request for the new URL.

You can configure GoodAccess Server to handle redirection responses (the default) or you can configure the server to return an HTTP response to the handheld that made the initial request, so that the handheld can send a new HTTP request with the new URL.

Setting GoodAccess Server to directly handle redirection responses reduces the time that would otherwise be spent in sending the Web server's response to the user handheld and followed by the user handheld returning its response to GoodAccess Server.

When GoodAccess Server is configured to handle redirection responses, each time it receives a redirection response from a Web server, it creates an HTTP request that uses the URL for the changed location and sends the request back to the Web server. When the Web server responds, GoodAccess Server returns the response to the browser. An exception to this sequence is when the Web server sends a redirection response and a *Set-Cookie* header and cookie handling is disabled on the server. In this case, to make sure the browser receives the cookie, GoodAccess Server sends the Web server's response directly to the browser.

When you enable GoodAccess Server to handle redirection responses, you can also configure it to limit the number of consecutive redirection responses it allows per response. This prevents the possibility of getting stuck in a redirection loop, which can occur if a series of redirection responses leads back to the original URL. For example:

URL A > URL B > URL C > URL A

If GoodAccess Server is set to allow only five redirection responses in a row, the loop in the example is followed once and then broken off before it gets to URL C the second time.

To set a limit on the number of consecutive redirection requests, you use the configuration property *redirect.max*.

The following table lists the properties for configuring GoodAccess Server to handle redirection responses:

Property Name	Default	Description
redirect.enable	<i>true</i>	If set to <i>true</i> , enables the server to handle redirection responses. If <i>false</i> , redirection responses are handled by the browser.
redirect.max	5	Maximum number of consecutive redirections allowed per request.
redirect.log.enable	<i>false</i>	If <i>true</i> , enables the server to log its redirection requests.

## Managing Cookies

To improve application performance, GoodAccess manages cookies on the server instead of sending them to destination devices for the handhelds to handle.

The server stores the cookies intended for handhelds in a local database. No cookies are sent to handhelds. When a handheld sends a request to a Web site, the server checks to see if it has a cookie for the site that is associated with the handheld. In this case, the server sends the cookie as part of the handheld's HTTP request.

Server-side cookies can improve application performance, since no over-the-air time is used for transmitting cookies. In addition, server-managed cookies save additional time for handhelds by following all HTTP redirects that include cookies and take care of Web logins requiring cookies.

Cookie data is stored in the server's database.

## Header Management

GoodAccess Server receives HTTP requests from a handheld and forwards them to their destination Internet hosts. Header management makes it possible for you to configure GoodAccess Server so that when it receives a handheld request intended for designated hosts it adds or modifies one or more HTTP headers in the request.

Header management settings are useful for accessing sites that are not configured to recognize GoodAccess (or other types of wireless) handhelds.

For example, the handheld might send GoodAccess as its user agent, but GoodAccess is not recognized by Hotmail. You can specify that the User-Agent header in all HTTP requests sent to *hotmail.com* is to be set to Mozilla/4.0, instead of GoodAccess.

You can use the default header-management property settings in the *config.props* file to change the User-Agent.

As a second example, you can add an email address header to all HTTP requests and set the header value to the email address of the handheld sending a request. When a Web server receives a request it can then use the handheld email address for billing or other purposes. Inserting a header for email addresses is provided through the default header-management property settings in the *config.props* file.

You specify headers, header values, and the hosts that receive them by associating each of these with a *header rule*. A header rule is defined through three properties:

- *header.rule\_name.header*
- *header.rule\_name.value*
- *header.rule\_name.hosts*

## Server Configuration Parameters

In each of these properties, *rule\_name* is replaced by the name assigned to the header rule. For example, in the default header properties, there is a header rule for assigning an email address header to all HTML requests. The rule name is *email* and correspondingly there are three *email* properties:

- *header.email.header*
- *header.email.value*
- *header.email.hosts*

**header.rule\_name.header.** This property specifies the name of the header to be inserted in an HTTP request or, if the header is already present, to be modified.

Example 1:

```
header.mozilla_ua.header User-Agent
```

In this example, the rule name is *mozilla\_ua* and the rule is set to apply to the User-Agent header.

Example 2:

```
header.email.header GI_EMAIL
```

In this example, the rule name is *email* and the rule is set to apply to a *GI\_EMAIL* header. Since *GI\_EMAIL* is not a standard HTTP header, GoodAccess Server will insert this header into requests.

**header.rule\_name.value.** This property specifies the value to be assigned to the header. The value is treated as a string and accepts variables representing either the value of a header (*%header\_name%*) or the value of a handheld making a request.

Example 1:

```
header.email.header GI_EMAIL
header.email.value %gi_emailaddress%
```



In this example, the *email* rule specifies that a *GI\_EMAIL* header is to be inserted in HTTP requests and that the value assigned to the header is to be the email address of the requesting handheld.

Example 2:

```
header.html_first.header Accept
header.html_first.value text/html, %Accept%
```

In this example, the rule *html\_first* assigns the Accept header the literal value *text/html*, followed by the received value of the Accept header. For example, if the Accept header in the request received from the handheld is:

```
Accept: text/vnd.wap.wml, text/plain, text/html
```

GoodAccess Server modifies the header to appear as:

```
Accept: text/html, text/vnd.wap.wml, text/plain, text/html
```

The modified header ensures that a Web server always sees *text/html* as the first element in the header.

Example 3:

```
header.mozilla_ua.header User-Agent
header.mozilla_ua.value Mozilla/4.0 (compatible; MSIE 6.0; Windows
NT 5.0; %User-Agent%)
```

In this example, the rule *mozilla\_ua* sets the User-Agent header to the literal value *Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0;* followed by the received value of the User-Agent header, followed by a close-parenthesis. For example, if the value of the User-Agent header sent by the handheld is:

```
GoodAccess 3.6.12 (PocketPC; GoodAccess; OS v. 4.4)
```

## Server Configuration Parameters

The header value sent to the destination Web server is:

*Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0;GoodAccess  
3.6.12 (PocketPC; GoodAccess; OS v. 4.4))*

This ensures that destination Web servers view the handheld as compatible with Mozilla/4.0, even if they do not recognize the user-agent string for GoodAccess, which identifies the handheld's user agent.

**header.rule\_name.hosts.** This property specifies the hosts that the rule applies to. If there are multiple hosts, enter them as a space-separated list.

At the beginning of a host name, you can use an asterisk (\*) as a wildcard to represent one or more components of the name. The following are all acceptable:

---

Example	Description
<i>header.rule1.hosts *</i>	The rule applies to all hosts
<i>header.rule2.hosts *.com</i>	The rule applies to all .com hosts
<i>header.rule3.hosts *.goodaccess.com</i>	The rule applies to all hosts that end with <i>goodaccess.com</i>

You cannot use an asterisk (\*) as a wildcard within a host name or as part of a component:

*// The following are not valid uses of '\*'*  
*header.bad\_rule1.hosts goodaccess.\*.com*  
*header.bad\_rule2.hosts \*oogle.com*

## Default Header-Management Properties

The default header-management properties include:

Property Name	Default	Description
header.enable	<i>false</i>	If set to <i>true</i> , enables use of header management.
header.email. header	<i>GI_EMAIL</i>	The header rule <i>email</i> applies to the header <i>GI_EMAIL</i> . Since <i>GI_EMAIL</i> is not a standard HTTP header, GoodAccess Server will insert this header in all HTTP requests sent to the hosts specified in <i>header.email.hosts</i>
header.email. value	<i>%gi_emailaddress%</i>	The header rule <i>email</i> sets the value of the <i>GI_EMAIL</i> header to the email address of the device sending a request.
header.email. hosts	<i>*</i>	The header rule <i>email</i> applies to HTTP requests sent to all hosts.
header.esn. header	<i>GI_ESN</i>	The header rule <i>esn</i> applies to the header <i>GI_ESN</i> . Since <i>GI_ESN</i> is not a standard HTTP header, GoodAccess Server will insert this header in all HTTP requests sent to the hosts specified in <i>header.esn.hosts</i>
header.esn. value	<i>%gi_esn%</i>	The header rule <i>esn</i> sets the value of the <i>GI_ESN</i> header to the ESN number of the device sending a request.
header.esn. hosts	<i>*</i>	The header rule <i>esn</i> applies to HTTP requests sent to all hosts.

## Server Configuration Parameters

Property Name	Default	Description
header.html_first. header	<i>Accept</i>	The header rule <i>html_first</i> applies to the HTTP Accept header.
header.html_first. value	<i>text/html, %Accept%</i>	The header rule <i>html_first</i> sets the value of the HTTP Accept header to <i>text/html</i> followed by the value of the Accept header as received from the handheld.  This has the effect of ensuring that the first element in the Accept header is always <i>text/html</i> .
header.html_first. hosts	<i>google.com</i>	The header rule <i>html_first</i> applies only to <i>google.com</i> . If <i>google.com</i> receives an HTTP Accept header where <i>text/vnd.wap.wml</i> is the first element, <i>google.com</i> returns a WML page.  The <i>html_first</i> rule ensures that <i>google.com</i> returns an HTML page to GoodAccess handhelds.
header.mozilla_ua. header	<i>User-Agent</i>	The rule <i>mozilla_ua</i> applies to the HTTP User-Agent header.

Property Name	Default	Description
header.mozilla_ua. value	<i>Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; %User-Agent%)</i>	The rule <i>mozilla_ua</i> sets the value of the User-Agent header to <i>Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0;</i> followed by the handheld user-agent string, followed by a close-parenthesis.
header.mozilla_ua. hosts	<i>hotmail.com google.com passport.com msn.com</i>	The rule <i>mozilla_ua</i> applies to <i>hotmail.com</i> , <i>google.com</i> , <i>passport.com</i> , and <i>msn.com</i> . These Web sites may not recognize the handheld's user-agent string ( <i>GoodAccess ...</i> ) but will see the handheld software as compatible with Mozilla/4.0 because of the value assigned through <i>header.mozilla_ua.value</i> .

# Differential Data

To reduce transmission time, GoodAccess Server supports differential-data transmission. If differential-data transmission is enabled, the server maintains a cache of the documents it sends to individual handhelds.

When a GoodAccess browser on the handheld sends an HTTP request for a document and the response comes back to GoodAccess Server, the server automatically checks its differential-data to see if the document is a newer version of a document previously sent to the handheld. If it is, the server compares the two versions and sends only the changes to the handheld.

A similar process happens when GoodAccess Server receives the updated version of a push message. Before sending it to the handheld, the server compares it to the previous version of the push message and sends only the updated portion of the message to the handheld.

When the GoodAccess handheld receives differential data from the server, it updates the document with new content, so both the server and the handheld have the latest versions. In general, sending differential data requires fewer packets and reduces transmission time.

The following table lists the properties for setting up differential-data transmission:

Property Name	Default	Description
diff.enable	<i>true</i>	If <i>true</i> , differential-data transmission is enabled.
diff.log.enable	<i>false</i>	If set to <i>true</i> , differential-data logging is enabled.

Property Name	Default	Description
<code>diff.cache.basedir</code>	<code>diff</code>	The directory under which documents previously sent to handhelds are cached. Documents are cached in binary form.
<code>diff.cache.maxsize</code>	<code>268435456</code>	The maximum size, in bytes, that the cache can reach before the server starts to delete older documents to make space for newer documents. The default is equivalent to 256 megabytes. The server deletes documents until the cache is reduced to the size specified by the <code>diff.cache.optsize</code> property.
<code>diff.cache.optsize</code>	<code>201326592</code>	The “optimal” cache size. The default is equivalent to 192 megabytes. This value should be set large enough to allow for an adequate number of previously cached documents and still leave room for adding new documents up to the maximum cache size specified.

The following is an excerpt from a differential-data log:

```
[date&time][Info]Diff Data Cache:Initializing Cache
[date&time][Info]Diff Data Cache:* Location:
C:/gaserver/GA38~1.15/diff
[date&time][Info]Diff Data Cache:* Scanning
[date&time][Info]Diff Data Cache:Found 20 buckets, 20 files, 42313 bytes
[date&time][Info]Differential Data:Diff not possible
[date&time][Info]Differential Data:Diff not possible
[date&time][Info]Differential Data:Diff not possible
[date&time][Info]Differential Data:*** Attempting to use diff data
[date&time][Info]Differential Data:Setting response key =
9e6a5be8c47e25afd428218fef7bf6d4
```

## Server Configuration Parameters

```
[date&time][Info]Differential Data:Cached new document with key =  
9e6a5be8c47e25afd428218fef7bf6d4  
[date&time][Info]Differential Data:Diff not possible  
[date&time][Info]Differential Data:*** Attempting to use diff data  
[date&time][Info]Differential Data:Setting response key =  
3de56d643e313a5b92c5e3d324760533  
[date&time][Info]Differential Data:Cached new document with key =  
3de56d643e313a5b92c5e3d324760533  
[date&time][Info]Differential Data:Browser says it accepts  
application/vnd.goodaccess.diff, proceeding with diff attempt  
[date&time][Info]Differential Data:Browser says it has old version of  
document with key = 3de56d643e313a5b92c5e3d324760533  
[date&time][Info]Differential Data:The server found the same version  
[date&time][Info]Differential Data:Content types are the same  
[date&time][Info]Differential Data:Diff successful, diff data sent
```

In the excerpt, the log is initialized and the location of the differential-data files is given as `C:/gaserver/GA37~1.15/diff` (equivalent to *install\_dir/diff*). At initialization, there are 20 buckets with a total of 20 files. The files take up 42313 bytes of disk cache.

A file holds a single document, such as an HTML page or a binary image. A bucket is essentially a directory that holds zero or more files and is used to distribute the files within the file system. When GoodAccess Server receives a document, it generates a unique key for the document. This key determines both the bucket and the name of the file in which the document is stored.

The files in the 20 buckets occupy 42313 bytes of disk cache. Given the default settings for *diff.cache.maxsize* (256 megabytes) and *diff.cache.optsize* (192 megabytes), there is a great deal of room for additional documents.



Following initialization, a new document arrives at the server, as indicated by the entries containing *Diff not possible* in the following sequence:

```
[date&time][Info]Differential Data:Diff not possible
[date&time][Info]Differential Data:Diff not possible
[date&time][Info]Differential Data:Diff not possible
[date&time][Info]Differential Data:*** Attempting to use diff data
[date&time][Info]Differential Data:Setting response key =
9e6a5be8c47e25afd428218fef7bf6d4
[date&time][Info]Differential Data:Cached new document with key =
9e6a5be8c47e25afd428218fef7bf6d4
```

Here, the document is new in the sense that it is not found in the differential-data cache. The document contains a text portion and two images. Otherwise the log would show only one entry with *Diff not possible*. Once the document has been identified as new, a unique ID (response key = 9e6a5be8c47e25afd428218fef7bf6d4) for the document is created. Any subsequent documents with the same content will generate the same key value and be compared with the cached document.

After the first document, a second new document arrives. As indicated by the single *Diff not possible*, this document contains text only, without referencing an image.

## Server Configuration Parameters

After the second new document is cached, a document arrives that matches a cached document. This results in the following sequence of entries:

```
[date&time][Info]Differential Data:Browser says it accepts  
application/vnd.goodaccess.diff, proceeding with diff attempt  
[date&time][Info]Differential Data:Browser says it has old version of  
document with key = 3de56d643e313a5b92c5e3d324760533  
[date&time][Info]Differential Data:The server found the same version  
[date&time][Info]Differential Data:Content types are the same  
[date&time][Info]Differential Data:Diff successful, diff data sent
```

In the first entry of the sequence, GoodAccess Server sees that the browser requesting a document accepts differential data. Next, the server confirms that the browser's document version matches a document cached on the server. Following this, the server confirms that the requested document and cached document have the same content types. Finally, the server computes and sends the difference between the browser's document and the requested document.

## Response Size

To minimize extensive, time-consuming downloads, you can enable and set a *responsesize* properties to limit the size of the content portion of an HTTP response that GoodAccess Server accepts for forwarding to a wireless handheld.

These properties include a maximum response size setting for various types of handhelds (Pocket PC, Palm OS, and others).

Setting with a *normal* suffix apply to data types that can be opened directly by the GoodAccess viewer. Examples include images, scripts, HTML, and WML (GDOM). The *attachment* setting applies to data types recognized by other applications on the handheld.

Examples include MS Excel documents, MS Word documents, and so on.

Response size limits apply directly to content. In the case of HTML content, it can be further limited by transformation properties.

The maximum applies to the number of bytes after compilation and any differential data optimization. If an HTTP response exceeds the response-size limit, the server sends an error to the viewer without sending the content.

Response size properties include:

Property Name	Default	Description
<code>responsesize.enable</code>	<i>true</i>	If set to <i>true</i> , enables the server to limit the content portion of an HTTP response.
<code>responsesize.PocketPC.normal</code>	<i>100000</i>	Specifies the maximum number of bytes of content that the server sends to a Pocket PC handheld.
<code>responsesize.PocketPC.attachment</code>	<i>1000000</i>	
<code>responsesize.PalmOS.normal</code>	<i>40000</i>	Specifies the maximum number of bytes of content that the server sends to a Palm OS handheld.
<code>responsesize.PalmOS.attachment</code>	<i>200000</i>	
<code>responsesize.default.normal</code>	<i>40000</i>	Specifies maximum content for all other handheld types not specifically listed.
<code>responsesize.default.attachment</code>	<i>40000</i>	
<code>responsesize.log.enable</code>	<i>false</i>	If set to <i>true</i> , enables logging of responses that are larger than the value set for <i>responsesize.max</i> .  Log entries contain the URL and size, in bytes, of responses that exceed the limit set by the property <i>responsesize.max</i> .

## Server Configuration Parameters

The following is an example of an entry in the response-size log:

967497215 [Info] Response Size Limiter: Document at <http://162.198.37.169:80/sites/obongo/wml/home.xml> is too large: 75662 bytes

## Push Support

This section enables the server to send push messages to handhelds. Also controls which hosts are allowed to initiate a push command. For more information, see “Setting Push Properties” on page 87.

## Server Console

Contains properties that control the GoodAccess Server Console. These settings are useful if you want to change the console port or reset login values.

The following table lists properties you can set to control console behavior.

---

Property Name	Default	Description
admin.html.enable	<i>true</i>	If set to <i>true</i> , enables use of the server console.
admin.log.enable	<i>true</i>	If set to <i>true</i> , enables the management log. This log records initialization of the server console and errors.  Use of the management log is not recommended in a production environment.

Property Name	Default	Description
admin.html.port	19001	<p>The port number the browser uses to access the server console.</p> <p>A browser accesses the server console through the following URL:  <i>http://hostname:port</i>            where <i>hostname</i> is the name of the host system running GoodAccess Server and <i>port</i> is the port number entered as the value of <i>admin.html.port</i>.</p>
admin.html.username	admin	<p>A user name for validating access to the server console.</p> <p>This is a required property. The server console requires a user name for authentication.</p>
admin.html.password	admin	<p>A password for validating access to the server console.</p> <p>This is a required property. The server console requires a password for authentication.</p>

# Traffic Display

Enables the GoodAccess Server Console to display the amount of data traffic sent to and from handhelds. These settings are useful for enabling the traffic display or changing the time interval for storing data.

The following table lists properties for enabling the traffic display and for setting a limit on the length of time traffic data is stored.

Property Name	Default	Description
trafficdisplay.enable	<i>true</i>	If set to <i>true</i> , enables the display of traffic between GoodAccess Server and handhelds.
trafficdisplay.maxage	7257600000	The length of time, in milliseconds, to store traffic data. The default is equivalent to 12 weeks.  When this property is commented out, and the server uses its own default, which is 12 weeks—the same as the default value of <i>trafficdisplay.maxage</i> . You can uncomment the property and set it to a different value to extend or reduce the default storage time.

## Log Management

The server automatically maintains production log files and makes available a number of diagnostic log files. In addition, it provides properties that allow you to specify the format used for displaying times in log files and to specify when and how often log files are rolled. This section describes log file properties you can set in the *config.props* file.

### Location of Log Files

All log files have a GoodAccess assigned name and are stored (by default) in the server's *install\_dir/logs* directory. **Note:** Filenames for individual logs (for example, *redirect.log*) are set by the server and cannot be changed.

### Log Rolling

The “Log rolling” section of the *config.props* file allows you to determine when and how frequently log files are rolled. When a log file is rolled, the file is closed and a new log file is opened with the current date and time appended to the log name.

Logs are renamed using the following syntax:

**name.log***MM-dd-yy.HH-mm-ss*

where *MM-dd-yy.HH-mm-ss* represents the time the log file roll operation occurred. For example, *access.log12-21-03.77-15-22*.

By default, log files are rolled every 24 hours, at midnight.

## Server Configuration Parameters

The following table describes the servers log rolling properties:

---

Property Name	Default	Description
log.roll.interval	<i>day</i>	Specifies the length of time that a log file is open, before it is closed and a new log file started. Log rolling intervals can be <i>none</i> , <i>hour</i> , or <i>day</i> .  If the value is set to <i>none</i> (or no value is specified) log rolling will not occur.

## Time Format Used in Time Stamps

Every entry in a log file has a time stamp marking the time that an event was logged.

## Configuring NT Event and Good Operations Logging

Use the following properties to configure the NT Event log and/or to send messages to the *good-ndc.log* file.

Properties for the NT Event log include:

---

Property Name	Default	Description
nthevent.log.enable	<i>true</i>	Enables the NT Event log. If <i>true</i> the log is enabled. This log stores severe errors encountered by the server, including server configuration errors.



Properties for the single log file include:

---

Property Name	Default	Description
good-ndc.log.enable	<i>true</i>	Enables a global log file for all communications between GoodAccess Server and the Good Operations Center. If <i>true</i> the log is enabled and all server communication related logging messages are stored in this file.

*Note:* In some cases, when communicating with the Good Operations Center, the GoodAccess Server and GoodLink Server share the same log messages. Even though the GoodAccess Server generates these messages, the term “GoodLink Server” may appear in the messages. For more information, see “Shared Log Messages” on page 52.

## About the Access Log

The access log records Web activity. Entries are recorded in an extended version of Common Log Format, a standard access-log format for Web servers. Each entry contains the following information:

---

Log Entry	Description
Remote ID	The handheld device identifier (email address) from which a wireless communication is received.
Date	The date and time of an HTTP request, enclosed in brackets. For example: <i>[28/Aug/2001:12:21:26 -0800]</i>
Request	<p>A request line, enclosed double quotes, exactly as it was received from the browser.</p> <p>A request line includes the destination URL and the method (usually GET) used to retrieve it. For example:</p> <p><i>"GET http://www.pozart.com:80/the/portal/cal/cday.po?date=20010828&amp;sk=zZ6Z2Q2t5IHVQgk82jTRgg-V HTTP/1.0"</i></p>
Status	A status code returned by the Web server, indicating success or a type of failure.
Bytes	The number of bytes of content returned by the Web server.
Referrer	The URL the browser was on before requesting the current URL. If no referrer-URL is specified, appears as - (minus sign).
User agent	<p>The type of browser used to make the request. <i>User agent</i> is enclosed in double quotes. For example: <i>"MyDevice 1.5 (GoodTechnology Handheld; OS v. 1.0)"</i></p> <p><i>User agent</i> is a standard extension to common log format.</p>

For security purposes, Good Technology recommends reviewing the access log periodically to monitor handheld connections on your network. The following is an example of an entry in an access log:

```
192.168.4.25 - - [27/Mar/2001:14:34:33 -0800] "GET http://localhost/prod/index.wml HTTP/1.0" 200 388 - "GoodAccess 8.1 (PPC; IP; OS v. 8.5 8192K)"
```

## SSL Support

The following properties are available for configuring Secure Socket Layer (SSL) communications (https) between the server and secure Web sites:

---

Property Name	Default	Description
ssl.enable	<i>true</i>	Enables SSL communications.
ssl.log.enable	<i>false</i>	If set to <i>true</i> , enables creation of an SSL log file.
ssl.trusted	<i>ssl/trusted.txt</i>	Location of the file that contains a list of installed certification authority root certificates.

# Advanced Configuration

The following properties are available for fine-tuning your server installation. Do not use these properties unless directed by Good Technical Support.

## HTTP Request and Response Management

### *Connections to a Web Server*

You can configure GoodAccess Server to time out if it cannot connect to a Web server within a specified length of time. You can also turn logging on to record problems in connecting to Web servers.

The time out settings is useful for preventing excessive wait times when a handheld tries to access a site supported by an overloaded or downed Web server. The logging setting is useful for troubleshooting Web server connection problems.

Properties for HTTP connections include:

Property Name	Default	Description
thread.min	50	The number of threads to allocate, prior to startup, for handling HTTP requests.
thread.max	200	The maximum number of concurrent threads to allow for handling HTTP requests.
thread.queue size	1000	The maximum number of HTTP requests the server will handle. The total includes active threads and requests in queue, waiting for connection threads from the thread pool.

---

Property Name	Default	Description
http.timeout	60000	The maximum length of time to wait for a connection to a Web server, in milliseconds. The default is equivalent to one minute.
http.localhost	###.###.###.###	The IP address or host name to be used in sending packets to a Web server. If you do not specify a <i>localhost</i> value, the server uses the default IP address of its host system.

## Outgoing Proxy Support

You can configure GoodAccess Server to use proxy servers for the HTTP and HTTPS (secure HTTP) requests it receives. There can be multiple servers for each protocol. For example, you can define a proxy server for all HTTP requests sent to *HTTP://www.oneurl.com* and a separate proxy for all HTTP requests sent to *HTTP://www.twourl.com*. Similarly, you can define separate proxy servers for HTTPS requests sent to the same destinations.

For each proxy server you define, you need to assign the proxy a proxy name and configure the following properties:

- *proxy.proxy\_name.host*
- *proxy.proxy\_name.port*
- *proxy.rule\_name.rule*

For a single proxy, there may be multiple rules, each with its own rule name.

For proxies that need to authenticate themselves in order to access an HTTP realm, you also need to set the following properties:

- *proxy.proxy\_name.realm.username*
- *proxy.proxy\_name.realm.password*

## Server Configuration Parameters

Proxies for HTTPS require an additional property:

*proxy.proxy\_name.options*

Thus, for two HTTP proxies, HTTP\_Proxy1 and HTTP\_Proxy2 and one HTTPS proxy, HTTPS\_Proxy1, you potentially have the following sets of properties:

---

<i>proxy.Http_Proxy1.host</i>	<i>proxy.Http_Proxy2.host</i>	<i>proxy.Https_Proxy1.host</i>
<i>proxy.Http_Proxy1.port</i>	<i>proxy.Http_Proxy2.port</i>	<i>proxy.Https_Proxy1.port</i>
		<i>proxy.Https_Proxy1.options</i>
<i>proxy.Http_Proxy1.realm.username</i>	<i>proxy.Http_Proxy2.realm.username</i>	<i>proxy.Https_Proxy1.realm.username</i>
<i>proxy.Http_Proxy1.realm.password</i>	<i>proxy.Http_Proxy2.realm.password</i>	<i>proxy.Https_Proxy1.realm.password</i>

In addition, for each proxy there must be one or more *proxy.rule\_name.rule* properties.

Properties for proxy server set up and logging include:

---

Property Name	Default	Description
<i>proxy.enable</i>	<i>false</i>	Enables GoodAccess Server to use a proxy server for making HTTP requests.
<i>proxy.log.enable</i>	<i>false</i>	Enables logging of proxy configuration information.
<i>proxy.proxy_name.host</i>	<hostname>   <IP address>	The host name or IP address of the proxy server identified by <i>proxy_name</i> . The identifier you set for <i>proxy_name</i> is used in subsequent properties
<i>proxy.proxy_name.port</i>	<port>	The port on which the named proxy receives requests.

Property Name	Default	Description
proxy. <i>proxy_name</i> . options	<i>tunnelssl</i>   <i>nonsecure</i>	<p>If the proxy named in the property is used for HTTPS, one of the following options, or a combination of the two options:</p> <p><i>tunnelssl</i></p> <p>The server tunnels SSL through the proxy to the Web server. GoodAccess Server opens an SSL connection to the Web server by means of the proxy. HTTPS requests and responses then go through the proxy, but are only decrypted at the Web server or GoodAccess Server not at the proxy.</p> <p><i>nonsecure</i></p> <p>Communication between GoodAccess Server and the proxy is carried out over a nonsecure connection, but communication between the proxy and the Web server is carried out over SSL.</p> <p>You can combine the two options, by entering both of them, separated by a space. In this case, GoodAccess Server tries the first option and, if this fails, it tries the second option.</p> <p>For example, to have the server attempt to establish an SSL tunneling connection, but switch to a nonsecure connection if this fails, you would enter:</p> <p><i>tunnelssl nonsecure</i></p> <p><b>Note:</b> To combine options, make sure you remove the “ ” separator between options that appears in the default properties file.</p>

Property Name	Default	Description
<code>proxy.rule_name. rule</code>	<code>&lt;protocol&gt;:// &lt;regex&gt; &lt;proxy_name&gt;</code>	<p>Names and defines a proxy rule. A proxy rule specifies a URL-matching pattern (a regular expression) and a proxy name.</p> <p>When the browser makes a request, the requested URL is matched against the patterns in all proxy rules and the rule with the most specific pattern is selected.</p> <p>The server accesses the URL via the proxy named in the selected rule. If the URL does not match any pattern or if the selected rule's proxy is the special keyword <i>noproxy</i> then no proxy is used and the server accesses the URL directly.</p> <p>The URL-matching pattern in a proxy rule must start with the protocol, either <i>HTTP://</i> or <i>HTTPS://</i></p> <p>For example, the following three rules might be used in an enterprise intranet environment where internet traffic is proxied through the firewall and intranet traffic is allowed a direct connection. The proxy for internet traffic is <b><i>publicproxy</i></b>.</p> <pre>proxy.http.rule http:/* publicproxy proxy.https.rule https:/* publicproxy proxy.intranet.rule http://corp.intranet noproxy</pre> <p>The following rules proxy a specific host that can't be accessed directly:</p> <pre>proxy.xyz_http.rule http://xyz.com proxyToXyz proxy.xyz_https.rule https://xyz.com proxyToXyz</pre>



Property Name	Default	Description
<b>proxy.proxy_name.</b> <b>realm.username</b>	<username>	<p>Specifies the user name that the named proxy needs to use for authentication when it accesses the HTTP realm entered for the <b>realm</b> component of the property name.</p> <p>For example, suppose Proxy server <i>proxy_1</i> is used for HTTP requests sent to <i>www.gaerver.com/pubs/docs</i>, the realm <i>gapubs</i> contains <i>www.gaerver.com/pubs/docs</i>, and the user name required for accessing <i>gapubs</i> is <i>pubsusr</i>. In this case, you would set <b>proxy.proxy_name.realm.username</b> as follows:</p> <p><i>proxy.proxy_1.gapubs.username</i>  <i>pubsusr</i></p>
<b>proxy.proxy_name.</b> <b>realm.password</b>	<password>	<p>Specifies the password that the named proxy needs to use for authentication when it accesses the HTTP realm entered for the <b>realm</b> component of the property name.</p> <p>Continuing the example above, if the password for <i>gapubs</i> were <i>pubpass</i>, you would set:</p> <p><i>proxy.proxy_1.gapubs.password</i>  <i>pubpass</i></p>

### *Proxy Log*

The proxy log records the proxy name assigned to a proxy server (*proxy.proxy\_name.host*) and port (*proxy.proxy\_name.port*) and lists each proxy rule (*proxy.rule\_name.rule*) assigned to the proxy.

The following are some examples of proxy log entries for a single proxy server.

```
[18/Jun/2002:13:43:18.136 PDT][Info]Proxy Manager: Proxy rcproxy=
192.168.1.1:8080 added
[18/Jun/2002:13:43:18.186 PDT][Info]Proxy Manager: Rule http://
www.rcdocs.com/* : rcproxy added
[18/Jun/2002:13:43:18.186 PDT][Info]Proxy Manager: Rule http://
www.goodaccess.com/public/* : rcproxy added
```

In the first entry, the proxy named *rcproxy* has been assigned to the proxy server at IP address 192.168.1.1, using port 8080.

The second and third entries show the proxy rules for using *rcproxy*. The rule in the second entry assigns all HTTP requests sent to *www.rcdocs.com* to *rcproxy*. The rule in the third entry assigns all requests to *rcproxy* that contain the URL *HTTP://www.goodaccess.com/public*.

# 8 Developer's Reference

---

## Application Support

When a wireless handheld requests an HTML page, the page sent in response to the request passes through two selection processes and is then configured according to the results of those processes.

- In the first process, GoodAccess Server attempts to identify the specific Web application the page is requested from. It uses the request URL and any POST data in the body of the request to make the identification. If a specific application can't be identified, it assigns the page to a *generic* application.
- In the second process, GoodAccess Server selects the page for transformation based on settings in the application's configuration file.

Once selected, a page can be transformed to reformat, truncate, and otherwise scale it down to make it suitable for viewing on the small screen of wireless handhelds (for example, removing images, unrolling tables, and truncating pages over a certain size).

GoodAccess Server includes a configuration file for the generic applications (*generic.props*). In addition, application developers can identify custom applications used by GoodAccess handhelds and creating separate configuration files for each custom application. In practice, this means:

1. Specifying the request URL and/or POST data to select the application.
2. For each application, specifying the request URLs, response URLs and/or POST data to select pages for transformation.
3. Specifying appropriate content transformations for each page.

You use the GoodAccess properties file (*config.props*) for Item 1 listed above and application-specific properties files for the other items.

### Benefits of this Approach

The application and page selection options provided by GoodAccess Server are robust. They enable application developers to fine tune transformations for very specific purposes.

For highly-structured applications and pages that include consistent sets of data, you can define and apply simple application selection and transformation criteria.

For robust applications that include a variety of data sets and page designs, you can define and apply highly-customized application and selection criteria.

For example:

- A simple application that looks up email addresses from a corporate directory, might work just fine with the settings already defined in the *generic* application properties file.
- A structured accounting application that extracts spreadsheet data from a single database, might need a customized application properties file that applies a single set of transformations to all pages in the application.
- A complex sales lead, order tracking system might need multiple application properties files, each with a large set of customized transformations that apply to individual, customized page types.

## Identifying the Application Requested

The GoodAccess Server properties file has one or more property sets to identify the application requested. One set is for a *generic* application—any application not identified specifically. The other sets are for specific applications (one set for each application). Properties included in each set are *app.name.selectors*, *app.name.properties*, and *app.name.reload*.

Sets of properties are identified by the *name* element they share. For example, *app.generic.selectors*, *app.generic.properties*, and *app.generic.reload* comprise the *generic* application set.

By default, the GoodAccess Server properties file comes with the properties set for *generic* applications and no others (*apps/generic.props*).

Properties to identify requested applications include:

Property Name	Default	Description
app. <i>name</i> .selectors	A list of one or more selectors.	<p>A list of one or more selectors for matching against the request URL and any POST data in the request.</p> <p>Each selector has a <i>requestUrl</i> component and a <i>postData</i> component.</p> <p>If one of the selectors in instance #1 of the property is a better match than any selector in any other instance of the property, the application identified by <i>name</i> in instance #1 is specified. Selector #1 is a better match than selector #2 if it is <i>narrower</i>—that is, if it allows for fewer possible matching URLs.</p> <p>For the syntax of selector components and the algorithm for comparing and ranking matches, see “Selector Component Syntax” on page 173.</p>
app. <i>name</i> .properties	app/ <i>name</i> .props	<p>Identifies the configuration file for the application identified by <i>name</i>. (The default application directory is <i>install_dir/app</i>.)</p>
app. <i>name</i> .reload	false	<p>Determines if properties for the application identified by <i>name</i> are reloaded when they change.</p> <p>If true, for every request that matches a selector in <i>app.name.selectors</i>, the timestamp of the properties file for <i>name</i> is compared to the last-loaded timestamp and the file is reloaded if it is newer.</p>

By default, the *generic.app* properties are:

```
app.generic.selectors requestUrl=.*
# app.generic.properties app/generic.props
# app.generic.reload false
```

The value of the *app.generic.selectors* property (*requestUrl=.\**) is a one-component selector equivalent to the following two-component selector:

```
(requestUrl=.* postData=.*)
```

- In the first property, the *postData* component is not required because the default for any component is “*.\**” (matches anything).
- Parentheses are used, when necessary, to group the components of a selector. The selector in the *app.generic.selectors* property doesn’t need parentheses because it contains only one component. If the alternative selector had no parenthesis, it would be interpreted as a list of two separate selectors (each with one component), with *postData=.\** as the first selector and *requestUrl=.\** as the second.
- The selector *requestUrl=.\** is “generic” in the sense that it is a match for any possible request. That is, each of its components (*explicit requestUrl* and *implicit postData*) match any possible item of its type in a request.
- Both components of the selector have to match for the selector to count as a match. If the selector was defined as *requestUrl=.\*//.\*/\*yourhost/\** and the request URL didn’t include *yourhost*, the selector would not be a match even though the implied *postData=.\** component was a match.
- The *app.name.properties* and *app.name.reload* properties have default values, so they are functional even if they are commented out. If you are creating a property set and the *app.name.properties* and *app.name.reload* properties use default values, you do not have to include them in the set.

The properties *app.name.selectors* and *app.name.properties* for a set work together as follows:

- If the browser's request URL and POST data match the *requestUrl* value and *postData* value of the property *app.name.selectors* for an application, this match causes the properties in the configuration file for that application (*app.name.properties*) to be applied to the response (i.e., to the page returned in response to the request).
- Because the *requestUrl* and *postData* values defined by *app.generic.props* are the broadest possible values (.\*), all request URLs or POST data will match those values. Therefore, if there is only one, *generic* set of properties defined, the configuration file *generic.props* applies to all requests. In effect, *generic* properties apply to any application requested.
- The properties in *generic.props* will be applied only if there is no match with a selector for the application. For example:
  - If you have provided a set of properties for a specific application,
  - Its selectors are narrower than .\*, and
  - The browser's request URL and POST data matches one of those selectors.

The properties in the configuration file for that application will be applied instead of the properties in *generic.props*.



## Selecting Pages and Providing Transformations

The configuration file *generic.props*, and any other configuration file you create, has one or more sets of *response.name.selectors* and *response.name.transforms* properties. Each property set is identified by the *name* element they share. These properties define how response pages are transformed.

In the default *generic.props* file, there is one set (*response.all.selectors* and *response.all.transforms*) that apply to all response pages.

---

Property Name	Default	Description
<i>response.name.selectors</i>	A list of one or more selectors.	<p>A list of one or more selectors for matching against the request URL, any POST data in the request, and the response URL. (The request URL and response URL may be two different URLs.)</p> <p>Each selector has three components: <i>requestUrl</i>, <i>postData</i>, <i>responseUrl</i></p> <p>If one of the selectors in the first instance of the property is a better, narrower match (i.e., allows for fewer possible matching URLs) than selectors in other instances of the property, the response identified by <i>name</i> in the first instance of the property is specified and the transformations listed in <i>response.name.transforms</i> are used.</p> <p>For the syntax of selector components and the algorithm for comparing and ranking matches, see “Selector Component Syntax” on page 173.</p>
<i>response.name.transforms</i>		Lists the content transformations. For a list of possible transformations, see the configuration file, <i>app/generic.props</i> .

A set of *response.name.selectors* and *response.name.transforms* properties work together as follows:

- A match is defined by comparing the *requestUrl* value and the URL of the request, the *postData* value and the POST data in the request, and the *responseUrl* value and the URL of the response page.
- If an application has multiple selectors defined in one or more instances of the *response.name.selectors* property, only the narrowest match is applied.
- A match causes the transformations for that response name to be applied to the response (i.e., to the page returned in response to the request).

For the *generic.props* file, the *response.all.transforms* property has the following values:

```
TransformIfLarge(size=6000)\
RemoveImgs\
CleanTables\
UnrollTables\
Truncate(size=3000)\
text="<Truncated by GoodAccess>")\
Stamp(text="<Reformatted by GoodAccess>")
```

*TransformIfLarge* is a switch that controls successive transformations listed below it in the file. By default, *TransformIfLarge* triggers these successive transformations only if the page is at least 6000 bytes.

Additional content transformations are listed at the end of the file. By default these transformations are commented out, but you can use any combination of the transformations in *generic.props* to build configuration files for specific applications.

## Selector Component Syntax

Syntax for selector components include:

Component Type	Example	Description
requestUrl	<i>requestUrl=.*//yourhost/.*</i>	A regular expression that is matched against the request URL.
postData	<i>postData=postVariable=Post \\+data</i>	<p>A regular expression that is matched against the POST data in the request.</p> <p><b>Note:</b> POST data is generally URL-encoded, with spaces replaced by plus sign (+). The regular-expression sequence representing literal plus sign is a backslash preceding the plus, as in \+. However, the backslash must be doubled in the properties file, so literal plus is written as \\+.</p>
responseUrl	<i>responseUrl=.*login.*</i>	A regular expression that is matched against the response URL. (The response URL differs from the request URL if the origin server redirects the request.)

The property *app.name.selectors* in the GoodAccess properties file uses only the requestUrl and postData components. The property *response.name.selectors* in the application configuration file uses all three components.

Note the following:

- Selector components are separated by a space. For example:

```
(requestUrl=.*URLPATTERN.* postData=.*Submit.*)
```

Another example:

```
(\
requestUrl=.*URLPATTERN.* \
postData=.*Submit.*\
)
```

In the second example, the space between the first component and the backslash (\) at the end of the first line separates the two components.

- A list with multiple selectors can select an application by both its DNS name and its IP address:

```
requestUrl=.*//mydomainname/* \
requestUrl=.*//1\.\2\.\3\.\4/*
```

- If an application is selected but the *response.name.selectors* property in its configuration file provides no matches, no transformations will occur. For this reason, consider copying the property set (*response.all.selectors* and *response.all.transforms*) from *generic.props* into configuration files for specific applications. This will apply the all-purpose generic transformations in these cases.

## Selector Matching and Ranking

The following algorithm is used to choose the best-matching selector:

- For application selectors, the two components are matched against the request URL and POST data. For response transformation selectors, the three components are matched against the request URL, POST data, and the response URL.
- A selector is ranked only when each of its components is a match.

- The highest ranking selector is chosen.

Although a selector can be ranked only if all its components match, ranking is by individual component. *requestUrls* are considered first, and *postData*s considered only if there is a tie in *requestUrls*. *responseUrls* are considered only for ranking response transformation selectors, and only if there is a tie in *postData*s.

Each component is ranked by narrowest match. Narrowness is determined by counting the number of literal characters in the regular expression: The more literal characters, the narrower the match. For ranking purposes, the regular-expression “character-set” is equivalent to a literal character. Thus *a.\** is narrower than *.\** but the four following expressions are equally narrow:

*“a”*, *“b.\*”*, *“.\*c”* *“\+.\*”*, *“[a-z].\*”*

- A tie in the selectors is broken by choosing the profile with the “alphabetically least” profile name. For example, with the following selectors:

*app.a.selectors requestURL=.\**  
*app.b.selectors requestURL=.\**

If the “a” and “b” profiles are the best matches, the “a” profile will always be preferred over “b.”

## Ignoring Case

If desired, you can set all response transformation selectors defined for a given application to ignore case. You can also set the regular expressions used to select applications to ignore case.

To set all response transformation selectors in an given application to ignore case, add the following property to the properties file for the application:

*response.selectors.ignorecase true*

For example, adding this property to the *apps/myapp.props* file causes all response transformation selectors defined for the *myapp* application to be case insensitive.

Similarly, you can set application profile selection on GoodAccess Server to ignore case. When the following property is set to *true* in the GoodAccess *config.props* file, the regular expressions used to select applications are case insensitive.

```
app.selectors.ignorecase true
```

## Best Practices for Application Properties

By default, the GoodAccess Server properties file comes with a single set of properties for generic applications (*apps/generic.props*) and no others. However, because GoodAccess Server can handle different applications with different transformations, Good Technology recommends using application-specific profiles. This means:

- Configuring the *config.props* file to have multiple *app.name.selectors* settings.
- Preparing multiple *apps/name.props* files.

Application-specific profiles are preferable to grouping all transformations into the single *apps/generic.props* file.

## Supported HTML Elements

The following table lists HTML Elements and attributes and indicates whether they are supported on GoodAccess handhelds. See also “HTML Color Support” on page 184.

Element	Attribute	Supported (Y/N)
A		Y
	HREF	Y
	NAME	N
ABBR		Y (always boldface)
ACRONYM		Y (always boldface)
ADDRESS		N
APPLET		N
AREA		N
B		Y
BASE		N
	HREF	N
	TARGET	N
BASEFONT		N
	SIZE	N
BDO		Y
BIG		Y
BLOCKQUOTE		N
BODY		Y
	ALINK	N
	BACKGROUND	N
	BGCOLOR	Y
	LINK	N
	ONLOAD	Y
	ONSCAN	N
	ONUNLOAD	N
	TEXT	N
	VLINK	Y

---

Element	Attribute	Supported (Y/N)
BR		Y
	CLEAR	N
BUTTON		Y
CAPTION		N
	ALIGN	N
CENTER		Y
CITE		Y (always boldface)
CODE		Y (always boldface)
COL		Y
COLGROUP		Y
DD		N
DEL		Y
DFN		Y (always boldface)
DIR		N
	COMPACT	N
DIV		Y
	ALIGN	Y
DL		N
	COMPACT	N
DT		N
EM		Y (mapped to bold)
FIELDSET		Y
FONT		Y
	COLOR	Y
	FACE	N
	SIZE	N
FORM		Y
	ACTION	Y
	METHOD	Y
	ENCTYPE	Y
	NAME	N
	ONRESET	N
	ONSUBMIT	Y



Element	Attribute	Supported (Y/N)
FRAME		N
FRAMESET		N
H1 - H6		Y
	ALIGN	Y
HEAD		Y
HR		Y
	ALIGN	N
	NOSHADE	N
	SIZE (1-5)	Y
	WIDTH	N
HTML		Y
I		Y (mapped to italic)
IFRAME		Y
IMG		Y
	ALIGN	Y
	ALIGN	Y
	ALT	Y
	BORDER	N
	HEIGHT	N
	HSPACE	Y (deprecated in HTML 4.01)
	ISMAP	N
	NAME	Y
	SRC	Y
	USEMAP	N
	VSPACE	Y (deprecated in HTML 4.01)
	WIDTH	N
INPUT TYPE=button		Y
	NAME	Y
	VALUE	Y
INPUT TYPE=checkbox		Y
	CHECKED	Y
	NAME	Y
	VALUE	Y

---

Element	Attribute	Supported (Y/N)
INPUT TYPE=file		N
INPUT TYPE=hidden		Y
	NAME	Y
	VALUE	Y
INPUT TYPE=image		(mapped to submit)
	ALIGN	N
	ALIGN	N
	ALT	N
	NAME	Y
	SRC	N
INPUT		Y
TYPE=password	MAXLENGTH	Y
	NAME	Y
	SIZE	Y
	VALUE	Y
INPUT TYPE=radio		Y
	CHECKED	Y
	NAME	Y
	SIZE	N
	VALUE	Y
INPUT TYPE=reset		N
	NAME	N
	VALUE	N
INPUT TYPE=scribble		N
	NAME	N
	SIZE	N
	VALUE	N
INPUT TYPE=submit		Y
	NAME	Y
	VALUE	Y

Element	Attribute	Supported (Y/N)
INPUT TYPE=text		Y
	MAXLENGTH	Y
	NAME	Y
	SIZE	Y
	VALUE	Y
INS		Y
ISINDEX		N
KBD		Y (always boldface)
LABEL		Y
LEGEND		Y
LI		Y
	TYPE	N
	VALUE	N
LINK		N
MAP		N
MENU		N
	COMPACT	N
META		Y
	CONTENT	Y
	HTTP-EQUIV	Y
	NAME	Y
NOFRAMES		N
NOSCRIPT		Y
OBJECT		N
OL		Y
	COMPACT	N
	START	N
	TYPE	N
OPTGROUP		Y
OPTION		Y
	SELECTED	Y
	VALUE	Y

Element	Attribute	Supported (Y/N)
P		Y
	ALIGN	Y
PARM		N
PRE		Y
	WIDTH	N
Q		Y
S		Y
SAMP		Y (always boldface)
SCRIPT		Y
	LANGUAGE	N (Javascript only)
	SRC	Y
	TYPE	N
SELECT		Y
	MULTIPLE	Y
	NAME	Y
	SIZE	Y
SMALL		N
SPAN		Y
STRIKE		N
STRONG		Y
STYLE		N
SUB		N
SUP		N
TABLE		Y
	ALIGN	N
	BGCOLOR	N
	BORDER	Y (including support for variable border width)
	BORDERCOLOR	N
	CELLPADDING	N
	CELLSPACING	N
	WIDTH	Y

Element	Attribute	Supported (Y/N)
TBODY		Y
TD		Y
	ALIGN	Y
	BGCOLOR	Y
	COLSPAN	Y
	HEIGHT	N
	NOWRAP	N
	ROWSPAN	N
	VALIGN	N
	WIDTH	Y
TEXTAREA		Y
	COLS	Y
	NAME	Y
	ROWS	Y
TFOOT		Y
TH		Y
	ALIGN	Y
	COLSPAN	Y
	HEIGHT	N
	NOWRAP	N
	ROWSPAN	N
	VALIGN	N
	WIDTH	Y
THEAD		N
TITLE		Y
TR		Y
	ALIGN	Y
	BGCOLOR	Y
	VALIGN	N
TT		N

---

Element	Attribute	Supported (Y/N)
U		Y
UL		Y
	COMPACT	N
	TYPE	N
VAR		Y (always boldface)

## HTML Color Support

GoodAccess Server has an HTML preprocessor that converts HTML color-specifying elements and/or attributes into other HTML tags and/or attributes. This “tidied” HTML is then converted into a Good Document Object Model (GDOM) object and is sent to the handheld as serialized GBDOM.

Color values may be named or may be represented in standard HTML RGB format: `xRrGgBb` is the color with red=`0xRr`, green=`0xGg`, blue=`0xBb`. The supported named colors are:

*black silver gray white maroon red purple fuchsia green lime olive yellow navy blue teal aqua*

The following table shows the currently supported color-specifying HTML elements and attributes and how they are encoded as a GBDOM element. In each case, the *color* is a supported name or hex value:

---

HTML	GDOM encoding
<code>&lt;font color="color"&gt;</code>	<code>&lt;span color=value&gt;</code>
<code>&lt;div style="color: color"&gt;</code>	<code>&lt;div color=value&gt;</code>
<code>&lt;div style="background-color: color"&gt;</code>	<code>&lt;div background-color=value&gt;</code>

## Unsupported HTML

The current server does not encode the following HTML elements:

---

### HTML

```
<body text="color">
<body link="color">
<body bgcolor="color">
<body vlink="color">
<body alink="color">
<table bgcolor="color">
<tr bgcolor="color">
<td bgcolor="color">
```

## Image support

Currently the handheld requests images by specifying the image URL and passing an *Accept* header that includes one or more image MIME types. The following graphics formats are discussed:

---

Acronym	Description	Type
GIF	Graphics Interchange Format	vector
JPEG	Joint Photographic Experts Group	raster
PNG	Portable Network Graphics	vector

The server converts from the source MIME type to the accepted MIME type using the Java Advanced Imaging (JAI) library. This library provides the ability to read/write various graphics formats and apply transformations. The following transformations are performed.

### *Image Transformations*

---

Source MIME Type	Destination MIME Type
BMP	JPEG
GIF	JPEG
JPEG	null
PNG	JPEG

**Important:** GoodAccess on the handheld cannot open a image file directly. The image must be included in an HTML image tag to render. For example:

```
<img>photo.bmp</img>
```



## Unsupported WML Features

With minor exceptions, GoodAccess handhelds support *Wireless Markup Language Specification Version 1.3*. The following table lists features of WML 1.3 that are either unsupported or (as allowed by the specification) unimplemented.

---

Element	Attribute	Unsupported/ Unimplemented	Comment
a	title	Unimplemented	
anchor	title	Unimplemented	
card	ordered	Unimplemented	Always treated as <i>true</i> .
do	optional	Unimplemented	Always treated as <i>false</i> .
fieldset		Unimplemented	Handheld ignores the <i>fieldset</i> element but displays all fields specified under <i>fieldset</i> .
go	accept-charset	Unsupported	If the origin server sends a WML deck containing <i>go accept-charset</i> , the handheld ignores it and sends back the list of character sets it supports: UTF-8 Latin 1 (ISO-8859-1) US ASCII

Element	Attribute	Unsupported/ Unimplemented	Comment
img	height	Unimplemented	
	width	Unimplemented	
	<i>vspace="percent"</i>	Unimplemented	If you enter <i>vspace</i> as a percentage, the handheld ignores it. If you enter <i>vspace</i> as a number of pixels, the handheld considers it but can not always follow it in displaying the image.
	<i>hspace="percent"</i>	Unimplemented	If you enter <i>hspace</i> as a percentage, the handheld ignores it. If you enter <i>hspace</i> as a number of pixels, the handheld considers it but can not always follow it in displaying the image.
	localsrc	Unsupported	The handheld always uses the URI specified in the <i>img src</i> attribute.
input	tabindex	Unimplemented	
	title=	Unimplemented	
meta	name	Unimplemented	When present, this meta element is ignored.
	forua	Unimplemented	Always treated as if <i>forua="true"</i> .
	scheme	Unimplemented	When present, this meta element is ignored.

Element	Attribute	Unsupported/ Unimplemented	Comment
optgroup		Unimplemented	The handheld ignores the <i>optgroup</i> element but displays all options specified under <i>optgroup</i> .
p	mode	Unimplemented	Paragraph text always wraps, regardless of <i>mode</i> .
select	tabindex	Unimplemented	
small		Unsupported	Appears as plain text in default size.

## Support for JavaScript

The table below lists JavaScript properties and functions that are supported by GoodAccess on the handheld.

### *Notation*

---

Y	Supported. Can be used in scripts.
I	Ignored. Scripts containing this will run, but the property or function return the empty string.
N	Not supported. Scripts containing this will not run beyond the point where this is used.
*	Support is built in to Nombas' ScriptEase library. Assumes full support for the ECMA Standard 262.
+	Disabled in the Nombas ScriptEase library for PDA devices like the Palm.
**	Not possible due to hardware constraints.
a	IE version string is "4.0 (compatible; MSIE 5.5; Windows NT 5.0; T312461)"
b	Device type is currently one of: "PalmOS," "PocketPC," "Desktop," or "Unknown."

### *JavaScript Support*

---

Item	Support (Y/N)	Notes
Array	Y*	
Boolean	Y*	
Date	N <sup>+</sup>	
Function	Y*	
Math	Y**	Partial: no float.
Number	Y*	
Object	Y*	
String	Y*	
RegExp	N <sup>+</sup>	

*JavaScript Support*

Item	Support (Y/N)	Notes
<b>Object events</b>		
onAbort	N	Very rarely used.
onBlur	N	
onClick	Y	Button, Image, Checkboxes, Radio Buttons.
onDbClick	N	
onChange	Y	Drop Down, Select, TextArea, Edit, Radio Button, Checkbox (complete).
onDragDrop	N**	Very rarely used.
onError	N	
onFocus	N	
onKeyDown	N	
onKeyPress	N	
onKeyUp	N	
onLoad	Y	In HTML <body>.
onMouseDown	N**	
onMouseMove	N**	
onMouseOut	N**	
onMouseOver	N**	
onMouseUp	N**	
onMove	N**	
onReset	N	
onResize	N**	
onSelect	N	

*JavaScript Support*

Item	Support (Y/N)	Notes
onSubmit	Y	Terminates a form submission if the handler explicitly returned <i>false</i> . If a return value was not specified, submission is allowed.
onUnload	N	
<b>Dialogs</b>		
alert	Y	
prompt	Y	Blocking dialog that returns a string.
confirm	Y	Blocking dialog that returns a boolean value.
<b>Document object</b>		
document.alinkColor	N**	
document.anchors	N	
document.applets	N	
document.bgColor	N**	
document.cookie	N	
document.domain	N?	
document.embeds	N	
document.fgColor	N**	
document.bgColor	Y	
document.forms[]	Y	
document.forms[].length	Y	Number of forms in doc.
document.{form_name}	Y	References form by its name.
document.{form}.action	Y	security: <a href="#">mailto:</a> & <a href="#">news:</a> not allowed.

*JavaScript Support*

Item	Support (Y/N)	Notes
document.{form}.elements[]	Y	
document.{form}.elements[].length	Y	
document.{form}.{element}.checked	Y	Checkbox & Radio Button
document.{form}.{element}.defaultChecked	I	Checkbox & Radio Button
document.{form}.{element}.defaultValue	I	Text & Password
document.{form}.{element}.defaultSelected	I	Option
document.{form}.{element}.form	Y	References parent form
document.{form}.{element}.length	Y	Select List or Drop Down
document.{form}.{element}.name	Y	
document.{form}.{element}.options[]	Y*	Read-only access.
document.{form}.{element}.selectedIndex	Y	Select List
document.{form}.{element}.type	I	
document.{form}.{element}.value	Y	Includes support for all relevant elements. This includes support for Drop Downs and Selects (both single and multi-select).
document.{form}.encoding	N	
document.{form}.method	Y	
document.{form}.name	Y	
document.{form}.target	N	
document.{form}.{elmt_name}[] (i.e. document.myform.category[i])	Y	Accesses the <i>i</i> th element having the specified name (for example, <i>category</i> ). Used for Checkboxes and Radio Buttons.

*JavaScript Support*

---

Item	Support (Y/N)	Notes
document.images[].border	N	
document.images[].complete	N	
document.images[].height	N	
document.images[].hspace	N	
document.images[].lowsrc	N	
document.images[].name	N	
document.images[].prototype	N	
document.images[].src	N	
document.images[].vspace	N	
document.images[].width	N	
document.lastModified	N	
document.layers	N	
document.linkColor	N**	
document.links[].hash	N	
document.links[].host	N	
document.links[].hostname	N	
document.links[].href	N	
document.links[].pathname	N	
document.links[].port	N	
document.links[].protocol	N	
document.links[].search	N	
document.links[].target	N	
document.links[].text	N	
document.plugins	N	
document.referrer	N	
document.title	Y	
document.URL	Y	
document.vLinkColor	N**	
document.captureEvents()	N	
document.close()	N	
document.{form}.{element}.blur()	N	



*JavaScript Support*

Item	Support (Y/N)	Notes
document.{form}.{element}.click()	N	
document.{form}.{element}.focus()	Y	
document.{form}.{element}.handleEvent()	N	
document.{form}.{element}.select()	N	
document.{form}.handleEvent()	N	
document.{form}.reset()	Y	
document.{form}.submit()	Y	
document.getSelection()	N	
document.handleEvent()	N	
document.images[].handleEvent()	N	
document.links[].handleEvent()	N	
document.open()	N	
document.releaseEvent()	N	
document.routeEvent()	N	
document.write()	N	
document.writeln()	N	

**Browser related objects**

navigator.appCodeName	Y	Returns "GoodAccess"
navigator.appName	Y	Returns "GoodAccess"
navigator.appVersion	Y	Returns IE version. <sup>a</sup>
navigator.language	Y	Returns "en"
navigator.mimeTypes	N	
navigator.platform	Y	Returns device type. <sup>b</sup>
navigator.plugins	N	
navigator.userAgent	Y	
navigator.javaEnabled()	Y	Returns <i>false</i> .
navigator.plugins.refresh()	N	
navigator.preference()	I	Disabled for security.
navigator.taintEnabled()	Y	Returns <i>false</i> .

*JavaScript Support*

Item	Support (Y/N)	Notes
<b>“this” object</b>		
Forms	Y	
Elements	Y	
<b>Window object</b>		
DnEvents	N?	IE only DnEvents.which=13
self	Y	
top	Y	Defaults to main window.
window.closed	Y?	
window.defaultStatus	I	Disabled for security.
window.document	Y	
window.frames	Y-	Mapped to ‘self’
window.history	N	History properties disabled for security.
window.innerHeight	I	
window.innerWidth	I	
window.length	I	
[String] window.location ( <i>accessing window.location as a String</i> )	Y	
window.location.hash	I	
window.location.host	I	
window.location.hostname	I	
window.location.href	Y	
window.location.pathname	I	
window.location.port	I	
window.location.protocol	I	
window.location.search	I	
window.locationbar	I	
window.menuubar	I	
window.name	Y-	Read only

*JavaScript Support*

Item	Support (Y/N)	Notes
window.opener	N	
window.outerHeight	I	
window.outerWidth	I	
window.pageXOffset	I	
window.pageYOffset	I	
window.parent	I	
window.personalbar	I	
window.scrollbars	I	
window.status	I	Disabled for security
window.statusbar	I	
window.toolbar	I	
window.window	Y	Current window
self.open()	Y	
top.open()	Y	
window.alert()	Y	
window.back()	Y	
window.blur()	I	
window.captureEvents()	I	
window.clearInterval()	I	
window.clearTimeout()	I	
window.close()	Y	
window.confirm()	Y	
window.disableExternalCapture()	I	
window.enableExternalCapture()	I	
window.find()	I	
window.focus()	Y	
window.forward()	Y	
window.handleEvent()	I	
window.history.back()	Y	
window.history.forward()	Y	
window.history.go()	Y-	Only for numerics.

*JavaScript Support*

---

Item	Support (Y/N)	Notes
window.home()	I	
window.location.reload()	Y	
window.location.replace()	Y	
window.moveBy()	I	
window.moveTo()	I	
window.open()	N	Currently not implemented.
window.print()	I	
window.prompt()	I	
window.releaseEvents()	I	
window.resizeBy()	I	
window.resizeTo()	I	
window.routeEvent()	I	
window.scroll()	I	
window.scrollBy()	I	
window.scrollTo()	I	
window.setInterval()	Y	Repeating timer.
window.setTimeout()	Y	Timer expiration set to specified value.
window.stop()	I	

## Using GoodInfo1.0 System Variables

For backward compatibility with GoodInfo 1.0, the following system variables let you submit hidden values from an application to GoodAccess Server. The system variables, which contain information unique to each handheld, are:

- *gi\_emailaddress*—Contains the email address of the handheld's user. You can use this variable to submit the user's email address, so the user does not need to enter it in a separate text field.

For example, you can use the *gi\_emailaddress* variable in an application that allows users to add their address to an email distribution list. The application might contain a menu of distribution lists and a submit button with the label "Add My Name." After the user chooses a list from the menu and then clicks the "Add My Name" button, the application submits the value of the *gi\_emailaddress* variable. (The value might be submitted to a CGI script on the server that manages the distribution lists.)

- *gi\_esn*—Contains the handheld's serial number. You can use this variable to submit the handheld's serial number. For example, you can use this variable to retrieve data on a server based on a handheld's serial number.
- The following variables contain the handheld's current time, date, and GMT offset:

*gi\_date\_month* (numeric value between 1 and 12)

*gi\_date\_day* (numeric value between 1 and 31)

*gi\_date\_year* (numeric value starting at 1999) (Set to current year by default.)

*gi\_time\_hour* (numeric value between 0 and 23) (24-hour clock)

*gi\_time\_min* (numeric value between 0 and 59)

*gi\_time\_sec* (numeric value between 0 and 59)

*gi\_time\_gmtoffset* (numeric value representing the number of hours off of GMT) (Can be positive or negative. If negative, the minus-sign "-" is not URL-encoded per RFC1738. The same applies to the "." if the value is a fraction.)

For any type of GoodAccess Application, you can use the system variables in an INPUT element with the "hidden" value for the Type attribute. Specify the system variable you want to use in the *\$(var\_name)* syntax for the Value attribute. For example:

```
<input type="hidden" name="email" value="$(gi_emailaddress)" />
```

# Index

---

## A

- Access log 50, 156
- adapters 128
- Admin log 50
- application development 93
  - customizing tips 119
  - deploying applications 113
  - process overview 96
  - testing and debugging 104
  - tools 9
- application properties files 99, 126, 165

## B

- bibliography, HTML and XML books 95

## C

- cache directory 24
- Cancel command 80
- client
  - configuration 124, 127
  - See also* handheld
- color support 184
- config.props file 37
- configuration
  - application support 124
  - client 124, 127
  - content conversion 124, 128
  - differential data 124, 144
  - header management 124, 137
  - host substitution 124, 130
  - HTTP requests and response 158
  - logging 125, 153

- push support 87, 125
- redirects 124, 134
- response size 125, 148
- server console 124, 150
- server cookies 136
- SSL support 125, 157
- traffic display 125
- URL substitution 124, 132
- configuration parameters 121
- content conversion 124
  - properties 51, 128
- content conversion
  - configuration 128
- Content log 50
- conventions, documentation xiii
- Converter log 51

## D

- data sources 2
- debugging tips, for application development 105
- Default log 50
- delivery states, for push messages 90
- diagnostic logs 51
- Diff log 51
- differential data 144
- differential-data transmission 144, 148
  - configuring 144
- documentation
  - conventions xiii
  - GoodAccess documentation set xiii

### E

encryption key 11

### G

Good Operations log 52

Good Technology

web site xiv

GoodAccess

accessing data sources 2

multiple servers 20

overview 1

system architecture 8

GoodAccess Server

backup and recovery 58

configuration 121

configuring 30

deploying applications 113

enabling users 41

installation 13

login and password 38

multiple 20

overview 9

property files 37

SSL certificates 29

starting 31

uninstalling 32

GoodAccess Server Console 1, 35

configuration 150

opening 38

GoodLink

integration with GoodAccess 10

software license agreement 24

GoodLink applications xi

GoodLink Data Center. *See* Good  
Operations Center.

GoodLink Management Console xi

GoodLink Management Server xi,  
25

GoodLink Network URL 25

GoodLink network, with

GoodAccess 8

GoodLink Server xi

installation 23

managing 67, 165

GoodLink software

installing on handhelds 62

### H

handhelds

installing GoodLink 62

preparing for GoodAccess 61

push messages 68

starting GoodAccess 63

uninstalling GoodAccess 65

header management 137

help

GoodAccess handheld xiii

GoodAccess Server xiii

host substitution 124, 130

host substitutions 55

host-substitution properties 131

HTML xii

HTML books 95

HTML elements, supported by  
GoodAccess 177

HTML support 177, 184

HTTP

connection properties 158

redirected requests 134

HTTP requests and response  
configuration 158

### I

image support 185

installation 13

accounts 21

customizing 30

GoodLink Server 23

installed files 29

requirements 14

SSL certificates 29

system requirements 16

Internet access, configuring 55

### J

Java 2 Platform xii

Java Server xii, 9

### L

license agreement 24

license key 15

Listen command 86

location

GoodLink Server software 25

log files



- access log 156
- configuration 153
- debugging applications 104
- directory 25, 52, 153
- Good Operations log 52
- response size log 150
- types of 50
- log rolling 153
- logging subsystem 9
- Logging tab 39, 49
- logging, overview 52, 153

## **M**

- Main tab 39, 40
- managing
  - GoodLink Servers 67, 165
- Microsoft Outlook (not installed) 16
- Microsoft Windows Server 2004 xi

## **N**

- notification states, push
  - messages 91
- NT Event log 49, 52

## **O**

- Outlook (not installed) 16

## **P**

- Palm OS handhelds
  - starting GoodAccess 65
- password, for GoodAccess
  - Server 38
- Pocket PC xi, 1
- Pocket PC handhelds
  - starting GoodAccess 65
- production logs 50
- properties
  - content conversion 51, 128
  - host substitution 131
  - HTTP connection 158
  - proxy server 160
  - redirection response 135
  - URL substitution 132
- properties file
  - for application development 97
- properties files 122
- proxy server properties 159, 160

- push commands 76
  - generating reports 84
  - invoking 76
- push database 89
- Push log 50
- push messages
  - about 67
  - cancelling 80
  - checking status 81
  - configuring hosts 87
  - database 89
  - delivery and notification
    - states 90
  - monitoring 86
  - on handhelds 68
  - process overview 69
  - sending 77
  - setting up 75
  - using 67
- push properties 87

## **R**

- Redirect log 51
- redirection response properties 134
- response size configuration 148
- response size log 150

## **S**

- security 11
  - encryption key 11
- server cookies 136
- server name 24
- server properties 37
- server registration 24
- Site Access tab 39, 54
- software license agreement 24
- SSL certificates 29
- SSL support 157
- Status command 82
- Stderr log 50
- Stdout log 50
- style sheet, creating custom 117
- Submit command 77
- synchronization
  - see also wireless synchronization
- system architecture 8
- system requirements 16

## Index

### T

- time stamp, in log files 154
- timeformat property 154
- traffic
  - display options 48
- Traffic tab 39, 47
- transformations 128, 171
  - built-in 100
  - custom 103, 114
  - enabling 129
  - introduction 93
- Treo 600 Smartphone xi

### U

- uninstalling, GoodAccess Server 32
- URL redirection 124
- URL substitution 124, 132
- Urlsubst log 51
- URL-substitution properties 132
- users
  - bulk allowing or denying 44
  - configuring Internet access 55
  - denying access 43
  - enabling on GoodAccess Server 41
  - exporting list of 45
  - preparing handhelds 61
  - viewing information on 42
- Users tab 39, 40

### W

- Web server connection 158
- Windows Mobile 2003 xi
- Windows NT account 26
- WML xii
  - unsupported features 187
- www.good.com xiv

### X

- XML and XSLT xii
- XML books 95