

Unified Access Control

Custom Sign-in Pages Solution Guide

Release



Published: 2010-05-18
Part Number: , Revision 1

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

This product includes the Envoy SNMP Engine, developed by Epilogue Technology, an Integrated Systems Company. Copyright © 1986-1997, Epilogue Technology Corporation. All rights reserved. This program and its documentation were developed at private expense, and no part of them is in the public domain.

This product includes memory allocation software developed by Mark Moraes, copyright © 1988, 1989, 1993, University of Toronto.

This product includes FreeBSD software developed by the University of California, Berkeley, and its contributors. All of the documentation and software included in the 4.4BSD and 4.4BSD-Lite Releases is copyrighted by the Regents of the University of California. Copyright © 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994. The Regents of the University of California. All rights reserved.

GateD software copyright © 1995, the Regents of the University. All rights reserved. Gate Daemon was originated and developed through release 3.0 by Cornell University and its collaborators. Gated is based on Kirton's EGP, UC Berkeley's routing daemon (routed), and DCN's HELLO routing protocol. Development of Gated has been supported in part by the National Science Foundation. Portions of the GateD software copyright © 1988, Regents of the University of California. All rights reserved. Portions of the GateD software copyright © 1991, D. L. S. Associates.

This product includes software developed by Maker Communications, Inc., copyright © 1996, 1997, Maker Communications, Inc.

Juniper Networks, Junos, Steel-Belted Radius, NetScreen, and ScreenOS are registered trademarks of Juniper Networks, Inc. in the United States and other countries. The Juniper Networks Logo, the Junos logo, and JunosE are trademarks of Juniper Networks, Inc. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

Products made or sold by Juniper Networks or components thereof might be covered by one or more of the following patents that are owned by or licensed to Juniper Networks: U.S. Patent Nos. 5,473,599, 5,905,725, 5,909,440, 6,192,051, 6,333,650, 6,359,479, 6,406,312, 6,429,706, 6,459,579, 6,493,347, 6,538,518, 6,538,899, 6,552,918, 6,567,902, 6,578,186, and 6,590,785.

Unified Access Control Deployment Scenarios Guide

Revision History
2010—Revised for UAC release 4.0.

The information in this document is current as of the date listed in the revision history.

END USER LICENSE AGREEMENT

READ THIS END USER LICENSE AGREEMENT ("AGREEMENT") BEFORE DOWNLOADING, INSTALLING, OR USING THE SOFTWARE. BY DOWNLOADING, INSTALLING, OR USING THE SOFTWARE OR OTHERWISE EXPRESSING YOUR AGREEMENT TO THE TERMS CONTAINED HEREIN, YOU (AS CUSTOMER OR IF YOU ARE NOT THE CUSTOMER, AS A REPRESENTATIVE/AGENT AUTHORIZED TO BIND THE CUSTOMER) CONSENT TO BE BOUND BY THIS AGREEMENT. IF YOU DO NOT OR CANNOT AGREE TO THE TERMS CONTAINED HEREIN, THEN (A) DO NOT DOWNLOAD, INSTALL, OR USE THE SOFTWARE, AND (B) YOU MAY CONTACT JUNIPER NETWORKS REGARDING LICENSE TERMS.

1. **The Parties.** The parties to this Agreement are (i) Juniper Networks, Inc. (if the Customer's principal office is located in the Americas) or Juniper Networks (Cayman) Limited (if the Customer's principal office is located outside the Americas) (such applicable entity being referred to herein as "Juniper"), and (ii) the person or organization that originally purchased from Juniper or an authorized Juniper reseller the applicable license(s) for use of the Software ("Customer") (collectively, the "Parties").

2. **The Software.** In this Agreement, "Software" means the program modules and features of the Juniper or Juniper-supplied software, for which Customer has paid the applicable license or support fees to Juniper or an authorized Juniper reseller, or which was embedded by Juniper in equipment which Customer purchased from Juniper or an authorized Juniper reseller. "Software" also includes updates, upgrades and new releases of such software. "Embedded Software" means Software which Juniper has embedded in or loaded onto the Juniper equipment and any updates, upgrades, additions or replacements which are subsequently embedded in or loaded onto the equipment.

3. **License Grant.** Subject to payment of the applicable fees and the limitations and restrictions set forth herein, Juniper grants to Customer a non-exclusive and non-transferable license, without right to sublicense, to use the Software, in executable form only, subject to the following use restrictions:

- a. Customer shall use Embedded Software solely as embedded in, and for execution on, Juniper equipment originally purchased by Customer from Juniper or an authorized Juniper reseller.
- b. Customer shall use the Software on a single hardware chassis having a single processing unit, or as many chassis or processing units for which Customer has paid the applicable license fees; provided, however, with respect to the Steel-Belted Radius or Odyssey Access Client software only, Customer shall use such Software on a single computer containing a single physical random access memory space and containing any number of processors. Use of the Steel-Belted Radius or IMS AAA software on multiple computers or virtual machines (e.g., Solaris zones) requires multiple licenses, regardless of whether such computers or virtualizations are physically contained on a single chassis.
- c. Product purchase documents, paper or electronic user documentation, and/or the particular licenses purchased by Customer may specify limits to Customer's use of the Software. Such limits may restrict use to a maximum number of seats, registered endpoints, concurrent users, sessions, calls, connections, subscribers, clusters, nodes, realms, devices, links, ports or transactions, or require the purchase of separate licenses to use particular features, functionalities, services, applications, operations, or capabilities, or provide throughput, performance, configuration, bandwidth, interface, processing, temporal, or geographical limits. In addition, such limits may restrict the use of the Software to managing certain kinds of networks or require the Software to be used only in conjunction with other specific Software. Customer's use of the Software shall be subject to all such limitations and purchase of all applicable licenses.
- d. For any trial copy of the Software, Customer's right to use the Software expires 30 days after download, installation or use of the Software. Customer may operate the Software after the 30-day trial period only if Customer pays for a license to do so. Customer may not extend or create an additional trial period by re-installing the Software after the 30-day trial period.
- e. The Global Enterprise Edition of the Steel-Belted Radius software may be used by Customer only to manage access to Customer's enterprise network. Specifically, service provider customers are expressly prohibited from using the Global Enterprise Edition of the Steel-Belted Radius software to support any commercial network access services.

The foregoing license is not transferable or assignable by Customer. No license is granted herein to any user who did not originally purchase the applicable license(s) for the Software from Juniper or an authorized Juniper reseller.

4. **Use Prohibitions.** Notwithstanding the foregoing, the license provided herein does not permit the Customer to, and Customer agrees not to and shall not: (a) modify, unbundle, reverse engineer, or create derivative works based on the Software; (b) make unauthorized copies of the Software (except as necessary for backup purposes); (c) rent, sell, transfer, or grant any rights in and to any copy of the Software, in any form, to any third party; (d) remove any proprietary notices, labels, or marks on or in any copy of the Software or any product in which the Software is embedded; (e) distribute any copy of the Software to any third party, including as may be embedded in Juniper equipment sold in the secondhand market; (f) use any 'locked' or key-restricted feature, function, service, application, operation, or capability without first purchasing the applicable license(s) and obtaining a valid key from Juniper, even if such feature, function, service, application, operation, or capability is enabled without a key; (g) distribute any key for the Software provided by Juniper to any third party; (h) use the

Software in any manner that extends or is broader than the uses purchased by Customer from Juniper or an authorized Juniper reseller; (i) use Embedded Software on non-Juniper equipment; (j) use Embedded Software (or make it available for use) on Juniper equipment that the Customer did not originally purchase from Juniper or an authorized Juniper reseller; (k) disclose the results of testing or benchmarking of the Software to any third party without the prior written consent of Juniper; or (l) use the Software in any manner other than as expressly provided herein.

5. **Audit.** Customer shall maintain accurate records as necessary to verify compliance with this Agreement. Upon request by Juniper, Customer shall furnish such records to Juniper and certify its compliance with this Agreement.

6. **Confidentiality.** The Parties agree that aspects of the Software and associated documentation are the confidential property of Juniper. As such, Customer shall exercise all reasonable commercial efforts to maintain the Software and associated documentation in confidence, which at a minimum includes restricting access to the Software to Customer employees and contractors having a need to use the Software for Customer's internal business purposes.

7. **Ownership.** Juniper and Juniper's licensors, respectively, retain ownership of all right, title, and interest (including copyright) in and to the Software, associated documentation, and all copies of the Software. Nothing in this Agreement constitutes a transfer or conveyance of any right, title, or interest in the Software or associated documentation, or a sale of the Software, associated documentation, or copies of the Software.

8. **Warranty, Limitation of Liability, Disclaimer of Warranty.** The warranty applicable to the Software shall be as set forth in the warranty statement that accompanies the Software (the "Warranty Statement"). Nothing in this Agreement shall give rise to any obligation to support the Software. Support services may be purchased separately. Any such support shall be governed by a separate, written support services agreement. TO THE MAXIMUM EXTENT PERMITTED BY LAW, JUNIPER SHALL NOT BE LIABLE FOR ANY LOST PROFITS, LOSS OF DATA, OR COSTS OR PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, OR FOR ANY SPECIAL, INDIRECT, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS AGREEMENT, THE SOFTWARE, OR ANY JUNIPER OR JUNIPER-SUPPLIED SOFTWARE. IN NO EVENT SHALL JUNIPER BE LIABLE FOR DAMAGES ARISING FROM UNAUTHORIZED OR IMPROPER USE OF ANY JUNIPER OR JUNIPER-SUPPLIED SOFTWARE. EXCEPT AS EXPRESSLY PROVIDED IN THE WARRANTY STATEMENT TO THE EXTENT PERMITTED BY LAW, JUNIPER DISCLAIMS ANY AND ALL WARRANTIES IN AND TO THE SOFTWARE (WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE), INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT DOES JUNIPER WARRANT THAT THE SOFTWARE, OR ANY EQUIPMENT OR NETWORK RUNNING THE SOFTWARE, WILL OPERATE WITHOUT ERROR OR INTERRUPTION, OR WILL BE FREE OF VULNERABILITY TO INTRUSION OR ATTACK. In no event shall Juniper's or its suppliers' or licensors' liability to Customer, whether in contract, tort (including negligence), breach of warranty, or otherwise, exceed the price paid by Customer for the Software that gave rise to the claim, or if the Software is embedded in another Juniper product, the price paid by Customer for such other product. Customer acknowledges and agrees that Juniper has set its prices and entered into this Agreement in reliance upon the disclaimers of warranty and the limitations of liability set forth herein, that the same reflect an allocation of risk between the Parties (including the risk that a contract remedy may fail of its essential purpose and cause consequential loss), and that the same form an essential basis of the bargain between the Parties.

9. **Termination.** Any breach of this Agreement or failure by Customer to pay any applicable fees due shall result in automatic termination of the license granted herein. Upon such termination, Customer shall destroy or return to Juniper all copies of the Software and related documentation in Customer's possession or control.

10. **Taxes.** All license fees payable under this agreement are exclusive of tax. Customer shall be responsible for paying Taxes arising from the purchase of the license, or importation or use of the Software. If applicable, valid exemption documentation for each taxing jurisdiction shall be provided to Juniper prior to invoicing, and Customer shall promptly notify Juniper if their exemption is revoked or modified. All payments made by Customer shall be net of any applicable withholding tax. Customer will provide reasonable assistance to Juniper in connection with such withholding taxes by promptly: providing Juniper with valid tax receipts and other required documentation showing Customer's payment of any withholding taxes; completing appropriate applications that would reduce the amount of withholding tax to be paid; and notifying and assisting Juniper in any audit or tax proceeding related to transactions hereunder. Customer shall comply with all applicable tax laws and regulations, and Customer will promptly pay or reimburse Juniper for all costs and damages related to any liability incurred by Juniper as a result of Customer's non-compliance or delay with its responsibilities herein. Customer's obligations under this Section shall survive termination or expiration of this Agreement.

11. **Export.** Customer agrees to comply with all applicable export laws and restrictions and regulations of any United States and any applicable foreign agency or authority, and not to export or re-export the Software or any direct product thereof in violation of any such restrictions, laws or regulations, or without all necessary approvals. Customer shall be liable for any such violations. The version of the Software supplied to Customer may contain encryption or other capabilities restricting Customer's ability to export the Software without an export license.

12. **Commercial Computer Software.** The Software is "commercial computer software" and is provided with restricted rights. Use, duplication, or disclosure by the United States government is subject to restrictions set forth in this Agreement and as provided in DFARS 227.7201 through 227.7202-4, FAR 12.212, FAR 27.405(b)(2), FAR 52.227-19, or FAR 52.227-14 (ALT III) as applicable.

13. **Interface Information.** To the extent required by applicable law, and at Customer's written request, Juniper shall provide Customer with the interface information needed to achieve interoperability between the Software and another independently created program, on payment of applicable fee, if any. Customer shall observe strict obligations of confidentiality with respect to such information and shall use such information in compliance with any applicable terms and conditions upon which Juniper makes such information available.

14. **Third Party Software.** Any licensor of Juniper whose software is embedded in the Software and any supplier of Juniper whose products or technology are embedded in (or services are accessed by) the Software shall be a third party beneficiary with respect to this Agreement, and such licensor or vendor shall have the right to enforce this Agreement in its own name as if it were Juniper. In addition, certain third party software may be provided with the Software and is subject to the accompanying license(s), if any, of its respective owner(s). To the extent portions of the Software are distributed under and subject to open source licenses obligating Juniper to make the source code for such portions publicly available (such as the GNU General Public License ("GPL") or the GNU Library General Public License ("LGPL")), Juniper will make such source code portions (including Juniper modifications, as appropriate) available upon request for a period of up to three years from the date of distribution. Such request can be made in writing to Juniper Networks, Inc., 1194 N. Mathilda Ave., Sunnyvale, CA 94089, ATTN: General Counsel. You may obtain a copy of the GPL at <http://www.gnu.org/licenses/gpl.html>, and a copy of the LGPL at <http://www.gnu.org/licenses/lgpl.html>.

15. **Miscellaneous.** This Agreement shall be governed by the laws of the State of California without reference to its conflicts of laws principles. The provisions of the U.N. Convention for the International Sale of Goods shall not apply to this Agreement. For any disputes arising under this Agreement, the Parties hereby consent to the personal and exclusive jurisdiction of, and venue in, the state and federal courts within Santa Clara County, California. This Agreement constitutes the entire and sole agreement between Juniper and the Customer with respect to the Software, and supersedes all prior and contemporaneous agreements relating to the Software, whether oral or written (including any inconsistent terms contained in a purchase order), except that the terms of a separate written agreement executed by an authorized Juniper representative and Customer shall govern to the extent such terms are inconsistent or conflict with terms contained herein. No modification to this Agreement nor any waiver of any rights hereunder shall be effective unless expressly assented to in writing by the party to be charged. If any portion of this Agreement is held invalid, the Parties agree that such invalidity shall not affect the validity of the remainder of this Agreement. This Agreement and associated documentation has been written in the English language, and the Parties agree that the English version will govern. (For Canada: Les parties aux présentes confirment leur volonté que cette convention de même que tous les documents y compris tout avis qui s'y rattache, soient rédigés en langue anglaise. (Translation: The parties confirm that this Agreement and all related documentation is and will be in the English language)).

Abbreviated Table of Contents

	Front Part	xvii
Part 1	Custom Sign-in Paages	
Chapter 1	Custom sign-in pages	3
Part 2	Index	
	Index	119

Table of Contents

	Front Part	xvii
	Related Documentation	xvii
	Document Conventions	xviii
	Requesting Technical Support	xviii
	Self-Help Online Tools and Resources	xix
	Opening a Case with JTAC	xix
Part 1	Custom Sign-in Pages	
Chapter 1	Custom sign-in pages	3
	Downloadable Zip Files	8
	Adding Custom Help Files	10
	Using the Template Toolkit	10
	How the Infranet Controller Uses the Template Toolkit	10
	Testing Your Custom Pages	11
	Understanding the Template Toolkit Language	13
	Template Comments	15
	Accessing and Updating Variables and Files	15
	GET Directive	15
	SET Directive	15
	getText Directive	15
	Creating Conditional Statements	19
	Conditional Operators	19
	IF and ELSIF Directives	20
	SWITCH and CASE Directives	20
	Creating Looping Constructs	21
	Unsupported Directives	21
	Getting Familiar with the Template Page Structure	21
	CGI Files	21
	Using Templates from the samples.zip File	22
	Customizing Look-and-Feel	23
	Required Components	23
	Displaying Custom Logos	23
	Simplifying the Code in the Sample Templates	24
	Version Directive	24
	Comment Section	24
	Header Tags	25
	Copyright and Footer Tags	25
	Page Banner	26
	Processing LoginPage.thtml	27

Using Cascading Style Sheets	28
Embedding CSS	29
Attaching a Standalone CSS	29
Customizing Error Handling and Navigation	35
Redirecting Based on Any and All Errors	35
Redirecting Based on Specific Errors	35
Defining Error Message Locations	35
Customizing Error Messages	35
Localizing Custom Sign-In Pages	36
Creating a Localized Set of Custom Sign-In Pages	37
Upgrade Considerations	37
Custom Templates Reference	38
<failedPolicy>	39
<failedPolicy>.name	39
<failedPolicy>.remediation	40
action	40
AgentInstall.shtml	40
AnonymousAuthentication	41
authenticate()	41
autocomplete	43
Cancel.shtml	43
ccInAcTimeout	44
ccRunning	44
CertificateAuthentication	45
changePasswordTitle	45
color	46
Continue	46
cval	46
Defender.shtml	47
delivery_mode	47
DoUnload()	47
DSLlaunchURL	48
ExceededConcurrent.shtml	48
FinishLoad()	49
focus	50
FriendlyTime	50
frmCasque	50
frmLogin	51
frmNextToken	51
frmSelectRoles	52
g_time	52
gCancelNewPinMode	52
gCancelNextTokenMode	53
geckoBrowser	53
GeneratePin.shtml	54
GetCookieValue()	54
GraceLoginsRemaining	55
GraceLoginUsed.shtml	55
HC_REMED_POLICIES_CHECK	56

HC_REMED_SHOW_ADV_PREFS	56
hclnAcTimeout	57
hcRunning	57
heading	58
help	58
help_on	58
HideRemed()	59
Home	59
ie_winxp	60
instructions	60
isInfranetAgent	60
isLinux	61
isSAMEnabled	61
keyboard.js	61
labelInstructions	62
listFailedPolicies	62
loading	63
locale	63
loginmode	63
Login()	64
LoginPage.shtml	65
LoginPageErrorArgs	65
LoginPageErrorCode	66
LoginPageErrorMessage	69
Logout.shtml	69
MinimumPasswordLength	71
msg	71
netacekey	71
netesecid_key	72
netesecid_loginmode	72
netesecid_pinerr	72
netesecid_realm	73
netesecid_username	73
netesecidactionCancel	74
netesecidactionEnter	74
Netscape4xx	74
NewPin.shtml	75
NextToken.shtml	76
onsubmit	76
p	76
PageErrorArgs	77
PageErrorCode	77
password	78
password	79
password2	79
PasswordChange.shtml	80
PasswordComplexityMessage	80
PasswordExpiration.shtml	81
PasswordExpirationFriendlyTime	81

PasswordExpirationSeconds	82
PasswordHistoryLength	82
pleasewait	82
PleaseWait.thtml	83
pleasewaitLogoutJSCode	84
PleaseWaitObjectEC	84
pleasewaitWin32	85
plugintype	85
portal	86
prompts	86
realm	87
realm	87
RealmList	88
recallLastRealmUsed()	88
Remediate.thtml	89
RoleList	90
runOnLoad	90
safari	90
secid_challenge	91
secidcontextid	91
secid_contextid	91
secid_loginmode	92
secid_pinerr	92
secid_pinformat	93
secid_pinselectmode	93
secid_systempin	93
secid_username	94
secidactionSavePin	94
secidpinformat	94
secidpinselectmode	95
SelectRole.thtml	95
setFailed()	95
setFinished	96
SetLastRealm()	96
setStarted()	97
setSucceeded()	97
setthankyou()	98
setup_classid	98
setup_codebase	98
Setupappversion	99
setup_codebase	99
showButtons	100
showChangePasswordTitle	100
showClose	101
showContinue	101
showHeading	102
showLoading	102
showRemedOption	102
showPolicies	103

ShowSystemPin.shtml	103
showTryAgain	103
signinAgain	104
signInLink	104
SM-NewPinSelect.shtml	105
SM-NewPinSystem.shtml	105
SM-NewUserPin.shtml	105
SM-NextToken.shtml	106
softid_error	106
softid_time	107
SSL.shtml	107
start_status	107
StartEP()	108
StartHC()	108
startPageLink	108
stopComponents()	109
SubmitClicked()	109
submitFrmCasque()	110
textClose	110
textContinue	111
textRemedOption	111
textTryAgain	112
thankyou()	112
TryAgain	113
tz_offset	113
upAndRunning()	113
welcome	114
win32	114

Part 2

Index

Index	119
-------	-----

List of Tables

	Front Part	xvii
	Table 1: IC Series Publications	xvii
	Table 2: Notice Icons	xviii
Part 1	Custom Sign-in Paages	
Chapter 1	Custom sign-in pages	3
	Table 3: Required Templates in Each Zip File	9
	Table 4: English Values of Common Variables	16
	Table 5: Conditional Operators	19
	Table 6: PIN usage indicators	54
	Table 7: Login Error Messages, Codes, and Related Notes	66
	Table 8: Additional Login Error Messages and Codes	70
	Table 9: PIN Error Messages	75
	Table 10: Additional PIN Error Messages	75
	Table 11: Page Error Codes	77
	Table 12: New PIN Assignment Messages	105

Front Part

- Related Documentation on page xvii
- Document Conventions on page xviii
- Requesting Technical Support on page xviii

Related Documentation

Table 1 on page xvii describes documentation for the IC Series Appliance.

All documentation is available at <http://www.juniper.net/techpubs/software/uac/>

Table 1: IC Series Publications

Book	Description
<i>UAC Interoperability with the ScreenOS Enforcer</i>	Details using the IC Series Appliance with the ScreenOS Enforcer.
<i>UAC Interoperability with the Junos Enforcer</i>	Details using the IC Series Appliance with the Junos Enforcer.
<i>Layer 2 and the IC Series RADIUS Server</i>	This guide provides an overview of RADIUS and the IC Series Appliance RADIUS server. Implementation details for 802.1X, MAC address authentication, and several other practical use cases are provided.
<i>UAC Guide to IF-MAP Federation</i>	Provides a comprehensive overview of IF-MAP and details using IF-MAP Federation with the UAC solution.
<i>Unified Access Control Administration Guide</i>	Includes comprehensive information about configuring the Unified Access Control solution and the Infranet Controller 4500 and 6500 appliances.
<i>Client Side Changes Guide</i>	Lists changes that the Infranet Controller clients and Odyssey Access Client make to client computers, including installed files and registry changes.
<i>Custom Sign-in Pages Solution Guide</i>	Includes guidelines for personalizing the look-and-feel of sign-in pages.
<i>Deployment Scenarios Guide</i>	Describes several example network scenarios for deploying the Unified Access Control solution.

Table 1: IC Series Publications (*continued*)

Book	Description
<i>Installation Guide</i>	Describes how to install the Infranet Controller 4500 and 6500 appliances on your network and begin configuration.
<i>Quick Start Guide</i>	Includes an example of configuring the Unified Access Control solution for a front-end server deployment scenario.
<i>Troubleshooting Guide</i>	Tips for troubleshooting the UAC Solution.

Document Conventions

Table 2 on page xviii defines notice icons used in this guide.

Table 2: Notice Icons

Icon	Meaning	Description
	Informational note	Indicates important features or instructions.
	Caution	Indicates a situation that might result in loss of data or hardware damage.
	Warning	Alerts you to the risk of personal injury or death.
	Laser warning	Alerts you to the risk of personal injury from a laser.

Requesting Technical Support

Technical product support is available through the Juniper Networks Technical Assistance Center (JTAC). If you are a customer with an active J-Care or JNASC support contract, or are covered under warranty, and need post-sales technical support, you can access our tools and resources online or open a case with JTAC.

- JTAC policies—For a complete understanding of our JTAC procedures and policies, review the JTAC User Guide located at <http://www.juniper.net/us/en/local/pdf/resource-guides/7100059-en.pdf> .
- Product warranties—For product warranty information, visit <http://www.juniper.net/support/warranty/> .
- JTAC hours of operation—The JTAC centers have resources available 24 hours a day, 7 days a week, 365 days a year.

Self-Help Online Tools and Resources

For quick and easy problem resolution, Juniper Networks has designed an online self-service portal called the Customer Support Center (CSC) that provides you with the following features:

- Find CSC offerings: <http://www.juniper.net/customers/support/>
- Search for known bugs: <http://www2.juniper.net/kb/>
- Find product documentation: <http://www.juniper.net/techpubs/>
- Find solutions and answer questions using our Knowledge Base: <http://kb.juniper.net/>
- Download the latest versions of software and review release notes: <http://www.juniper.net/customers/csc/software/>
- Search technical bulletins for relevant hardware and software notifications: <https://www.juniper.net/alerts/>
- Join and participate in the Juniper Networks Community Forum: <http://www.juniper.net/company/communities/>
- Open a case online in the CSC Case Management tool: <http://www.juniper.net/cm/>

To verify service entitlement by product serial number, use our Serial Number Entitlement (SNE) Tool: <https://tools.juniper.net/SerialNumberEntitlementSearch/>

Opening a Case with JTAC

You can open a case with JTAC on the Web or by telephone.

- Use the Case Management tool in the CSC at <http://www.juniper.net/cm/> .
- Call 1-888-314-JTAC (1-888-314-5822 toll-free in the USA, Canada, and Mexico).

For international or direct-dial options in countries without toll-free numbers, see <http://www.juniper.net/support/requesting-support.html> .

PART 1

Custom Sign-in Paages

- Custom sign-in pages on page 3

CHAPTER 1

Custom sign-in pages

The customizable sign-in pages feature enables you to personalize the look-and-feel of the pre-authentication, password management, and Secure Meeting pages that the presents to administrators and end-users. The tasks you can accomplish include:

- Customizing the presentation style using CSS or table-based design
- Customizing error message text
- Redirecting traffic to non-Infranet Controller pages based on error codes
- Localizing custom sign-in pages in supported languages
- Adding custom HTML help

The Infranet Controller appliance ships with several complete sets of sample custom sign-in pages that you can modify to suit your particular look-and-feel and behavior. You can download these files from the Infranet Controller. This document describes the files, their uses, and how to modify them. The Infranet Controller contains these files in .zip file format. You can download the zip files from the **Authentication > Signing In > Sign-in pages > Upload custom sign-in pages** page.



NOTE: The files described in this document apply only to agentless configurations, except for *AgentInstall.html* and *PleaseWait.html*, which are used to install the Odyssey Access Client.

In this guide and other documentation for the unified access control solution, the names Odyssey Access Client and *Host Checker* are used to refer to the specific type of UAC Agent.

You can achieve a good understanding of the process of modifying and uploading your custom sign-in pages by reading through this document completely before attempting to perform the customization. When you feel you have a comprehensive understanding of what is involved, you should consider downloading the **samples.zip** file and modifying one or two pages first, just to test your understanding. There is a strong likelihood that you will want to apply a similar look and feel to all of the pages, which you can accomplish fairly simply, if you follow the directions provided in this document. Other tasks, such as redirecting traffic based on errors may require more skill and time.

Custom sign-in pages can potentially require a large amount of memory and disk space. In order to provide custom sign-in pages per IVS, we recommend you customize the sample custom sign-in pages provided with the Infranet Controller appliance.

You can find the following topics in this document:

- Downloadable Zip Files on page 8
- Adding Custom Help Files on page 10
- Using the Template Toolkit on page 10
- Using Templates from the samples.zip File on page 22
- Customizing Error Handling and Navigation on page 35
- Localizing Custom Sign-In Pages on page 36
- Upgrade Considerations on page 37
- Custom Templates Reference on page 38
- <failedPolicy> on page 39
- <failedPolicy>.name on page 39
- <failedPolicy>.remediation on page 40
- action on page 40
- AgentInstall.shtml on page 40
- AnonymousAuthentication on page 41
- authenticate() on page 41
- autocomplete on page 43
- Cancel.shtml on page 43
- ccInAcTimeout on page 44
- ccRunning on page 44
- CertificateAuthentication on page 45
- changePasswordTitle on page 45
- color on page 46
- Continue on page 46
- cval on page 46
- Defender.shtml on page 47
- delivery_mode on page 47
- DoUnload() on page 47
- DSLaunchURL on page 48
- ExceededConcurrent.shtml on page 48
- FinishLoad() on page 49
- focus on page 50
- FriendlyTime on page 50
- frmCasque on page 50

- frmLogin on page 51
- frmNextToken on page 51
- frmSelectRoles on page 52
- g_time on page 52
- gCancelNewPinMode on page 52
- gCancelNextTokenMode on page 53
- geckoBrowser on page 53
- GeneratePin.shtml on page 54
- GetCookieValue() on page 54
- GraceLoginsRemaining on page 55
- GraceLoginUsed.shtml on page 55
- HC_REMED_POLICIES_CHECK on page 56
- HC_REMED_SHOW_ADV_PREFS on page 56
- hcInAcTimeout on page 57
- hcRunning on page 57
- heading on page 58
- help on page 58
- help_on on page 58
- HideRemed() on page 59
- Home on page 59
- ie_winxp on page 60
- instructions on page 60
- isInfranetAgent on page 60
- isLinux on page 61
- isSAMEnabled on page 61
- keyboard.js on page 61
- labelInstructions on page 62
- listFailedPolicies on page 62
- loading on page 63
- locale on page 63
- loginmode on page 63
- Login() on page 64
- LoginPage.shtml on page 65
- LoginPageErrorArgs on page 65
- LoginPageErrorCode on page 66
- LoginPageErrorMessage on page 69
- Logout.shtml on page 69

- MinimumPasswordLength on page 71
- msg on page 71
- netacekey on page 71
- netesecid_key on page 72
- netesecid_loginmode on page 72
- netesecid_pinerr on page 72
- netesecid_realm on page 73
- netesecid_username on page 73
- netesecidactionCancel on page 74
- netesecidactionEnter on page 74
- Netscape4xx on page 74
- NewPin.shtml on page 75
- NextToken.shtml on page 76
- onsubmit on page 76
- p on page 76
- PageErrorArgs on page 77
- PageErrorCode on page 77
- password on page 78
- password on page 79
- password2 on page 79
- PasswordChange.shtml on page 80
- PasswordComplexityMessage on page 80
- PasswordExpiration.shtml on page 81
- PasswordExpirationFriendlyTime on page 81
- PasswordExpirationSeconds on page 82
- PasswordHistoryLength on page 82
- pleasewait on page 82
- PleaseWait.shtml on page 83
- pleasewaitLogoutJSCode on page 84
- PleaseWaitObjectEC on page 84
- pleasewaitWin32 on page 85
- plugintype on page 85
- portal on page 86
- prompts on page 86
- realm on page 87
- realm on page 87
- RealmList on page 88

- `recallLastRealmUsed()` on page 88
- `Remediate.html` on page 89
- `RoleList` on page 90
- `runOnLoad` on page 90
- `safari` on page 90
- `secid_challenge` on page 91
- `secidcontextid` on page 91
- `secid_contextid` on page 91
- `secid_loginmode` on page 92
- `secid_pinerr` on page 92
- `secid_pinformat` on page 93
- `secid_pinselectmode` on page 93
- `secid_systempin` on page 93
- `secid_username` on page 94
- `secidactionSavePin` on page 94
- `secidpinformat` on page 94
- `secidpinselectmode` on page 95
- `SelectRole.html` on page 95
- `setFailed()` on page 95
- `setFinished` on page 96
- `SetLastRealm()` on page 96
- `setStarted()` on page 97
- `setSucceeded()` on page 97
- `setthankyou()` on page 98
- `setup_classid` on page 98
- `setup_codebase` on page 98
- `Setupappversion` on page 99
- `setup_codebase` on page 99
- `showButtons` on page 100
- `showChangePasswordTitle` on page 100
- `showClose` on page 101
- `showContinue` on page 101
- `showHeading` on page 102
- `showLoading` on page 102
- `showRemedOption` on page 102
- `showPolicies` on page 103
- `ShowSystemPin.html` on page 103

- showTryAgain on page 103
- signinAgain on page 104
- signInLink on page 104
- SM-NewPinSelect.shtml on page 105
- SM-NewPinSystem.shtml on page 105
- SM-NewUserPin.shtml on page 105
- SM-NextToken.shtml on page 106
- softid_error on page 106
- softid_time on page 107
- SSL.shtml on page 107
- start_status on page 107
- StartEP() on page 108
- StartHC() on page 108
- startPageLink on page 108
- stopComponents() on page 109
- SubmitClicked() on page 109
- submitFrmCasque() on page 110
- textClose on page 110
- textContinue on page 111
- textRemedOption on page 111
- textTryAgain on page 112
- thankyou() on page 112
- TryAgain on page 113
- tz_offset on page 113
- upAndRunning() on page 113
- welcome on page 114
- win32 on page 114

Downloadable Zip Files

Although each zip file contains all the templates, each zip file is for a particular set of features:

- **Sample.zip**—This zip file is for standard Infranet Controller pages, including standard Infranet Controller pre-authentication pages, ACE pre-authentication pages, ACE pre-authentication pages for use with eTrust SiteMinder, and password management pages.
- **SoftID.zip**—This zip file is for use with the RSA Soft ID client.
- **Kiosk.zip**—This zip file is for use by kiosk users or for any system in which you want to lock out keyboard-based login.

Certain templates are required when uploading the zip files. If you do not include these required templates, a warning message appears. “Custom sign-in pages” on page 3 lists the required templates for each zip file.

Table 3: Required Templates in Each Zip File

	Samples.zip	Kiosk.zip	SoftID.zip (ACE)
Cancel.thtml	R	R	R
Defender.thtml	R		
ds.css			
ExceededConcurrent.thtml	R	R	R
GeneratePin.thtml	R	R	R
GraceLoginUsed.thtml	R		
keyboard.js		R	
LoginPage.thtml	R	R	R
Logout.thtml	R	R	R
NewPin.thtml	R	R	R
NextToken.thtml	R	R	R
PasswordChange.thtml	R		
PasswordExpiration.thtml	R		
PleaseWait.thtml	R	R	R
Remediate.thtml	R	R	
SelectRole.thtml	R		R
ShowSystemPin.thtml	R	R	
SM-NewPinSelect.thtml	R		
SM-NewPinSystem.thtml	R		
SM-NewUserPin.thtml	R		
SM-NextToken.thtml	R		

Table 3: Required Templates in Each Zip File (*continued*)

	Samples.zip	Kiosk.zip	SoftID.zip (ACE)
SSL.shtml	R	R	R

To customize Infranet Controller pages, you must use the Template Toolkit language. For more information, see “Understanding the Template Toolkit Language” on page 13.

For information on how to internationalize custom sign-in pages, see “Localizing Custom Sign-In Pages” on page 36.

Adding Custom Help Files

You can add custom help files by creating your own HTML help files and linking them to the custom sign-in pages. You can easily add links to the basic HTML in the custom sign-in pages.

If you add customized online Help to your custom sign-in pages, you must set the destination URL to point to a **.html** file, not a **..shtml** file. For example, in the following input statement, the help file, **help.html**, should not have a **..shtml** extension:

```
<input type='submit' name='help' value="Help"
onclick='window.open("<% Home%>/imgs/help.html",
"wndHelp","height=400,width=500,resizeable=yes,scrollbars=yes");
return false;'>
```

If you use the **..shtml** extension, the **Help** link on the custom sign-in page returns an error.

Using the Template Toolkit

The Template Toolkit is an open source software package written in Perl, that enables you to build and replicate Web pages using a special-purpose language. The Template Toolkit language is simple and easy to learn.

How the Infranet Controller Uses the Template Toolkit

The Infranet Controller system software recognizes the Template Toolkit directives and is pre-configured to process templates. The Infranet Controller processes templates into **.html** files, which it then displays as administrator Web console pages or as end-user Web pages. You cannot change all of the generated Web pages, but you can customize the ones noted in this topic. You can also modify a variety of presentation items on the **UI Options** page in the administrative Web console.

The Template Toolkit processing utilities transform template source files into complete HTML files. The source files are a combination of HTML and Template Toolkit directives. The source files carry a user-defined extension, but the Infranet Controller recognizes the file extension of **..shtml**. Processed files typically carry a **.html** file extension.



NOTE: When creating customized sign-in pages, save them as UTF-i.

Testing Your Custom Pages

There are two ways to test your custom pages:

- Using the Template Toolkit utilities you have installed locally.
- Uploading customized files to the Infranet Controller.

If you are customizing just one or two of the sample template files, you might find it easiest to upload the files to the Infranet Controller. If you do not have access to a test system, or if you are customizing many of the sample template files or making significant changes to those files, you may want to install and configure the Template Toolkit on your local machine.

Setting up the Template Toolkit Locally

You can install the Template Toolkit on your local system. Because it is a Perl application, you need a Perl distribution on your system before you install the Template Toolkit. If you run a Linux or Mac OS X system, you probably already have a Perl distribution on your system. If you run a Windows system, you may need to install the ActiveState ActivePerl distribution, which you can find at <http://www.activestate.com/>.

If you prefer, you can install Cygwin, which allows you to run UNIX programs in a virtual environment on your Windows system. You can download Cygwin from <http://sourceware.org/cygwin/>.

It is likely that you will need to download the latest version of the Perl AppConfig module, which allows your Perl applications to read and process configuration files. You can find the latest version of AppConfig at <http://www.cpan.org/modules/by-module/AppConfig/>.

You can find a **gzip-tar** archive of the latest version of the Template Toolkit at <http://www.cpan.org/modules/by-module/Template/>.

Follow these steps:

1. Download, install, and configure Perl, ActivePerl, or Cygwin, depending on the type of operating system you are running, and your personal requirements.
2. Download, install, and configure AppConfig.
3. Download, install, and configure the Template Toolkit.



NOTE: The Template Toolkit distribution includes a number of useful documents. You should print the INSTALL document and read it closely. You must perform a number of configuration steps as detailed in the INSTALL document. If you try to perform the steps as described in INSTALL, but have not performed the requisite configuration, the installation fails. Fortunately, you can go back and perform the configuration steps you missed or performed incorrectly, and try to install again, until the installation succeeds.

After you have successfully installed and configured the Template Toolkit, you can generate HTML files from the sample templates that are available on the Infranet Controller. The Infranet Controller is pre-configured to process Template Toolkit files and many of the Infranet Controller components are not available on your local system. Therefore, you are unable to run the Template Toolkit in exactly the same way or to obtain the same exact results. However, you can simulate the processing closely enough to test the presentation. You will encounter the following limitations:

- You can only test the presentation of the sample templates on your local system. You cannot test the actual pre-authentication logic unless you upload the files to an Infranet Controller system.
- You must do one of the following:
 - Use the Template Toolkit [% TAGS %] directive to change any new directives so that they use the angle bracket tokens (<% %>) used in the Juniper Networks sample templates.
 - Use the standard Template Toolkit tokens ([% %]) and when satisfied with the presentation, modify them to angle bracket tokens (<% %>) before uploading to the Infranet Controller.
 - The Template Toolkit provides two command-line utilities for processing template files: *tpage*, which enables you to process a single template at a time, and *ttree*, which enables you to process an entire directory of template files. Although you can use the *ttree* utility, the Infranet Controller uses a proprietary process that simulates the *tpage* utility, and so, you may find your final results more effective by using *tpage* only.

Downloading Sample Templates and Uploading Files to the Infranet Controller

Your other option for testing customized template files is to upload the files to the Infranet Controller. Although this may be a bit tedious, if you are only customizing a handful of pages, you might find it easier than setting up the Template Toolkit environment on your local system.



NOTE: The total combined size of all uploaded customizable zip files can not exceed 12MB.

To make sure you can view the customized pages, you must perform these procedures.

To download the sample templates:

1. Sign in to the Infranet Controller as an administrator.
2. Select **Signing In > Sign In > Sign-in** pages.
3. Click **Upload Custom Pages**.
4. Download one or more of the sample template files.
5. Unzip the files on your local system.

To customize and upload the new templates:

1. Customize the files, as you like.
2. Add the files to a new zip file.



NOTE: When creating your zip file, zip only the template files. Do not include the folder that contains the templates in the zip file.

3. On the **Upload Custom Sign-In Pages** page, enter a name for the template files. This is a label that is used to identify the pages when you associate them with a specific URL.
4. Select the page type.
5. Click **Browse** to locate the new zip file containing your customized files.
6. Click **Upload Custom Pages**.

If you want your end-users to be able to see the new templates, you need to associate the templates, using the label you assigned to them, with a URL that you specify for a given role.

To associate the templates with a role-specific URL:

1. Select **Signing In > Sign-in > Sign-in Policies**.
2. Click **New URL**.
3. Configure the new policy and URL, using the sign-in policies instructions in the Administration Guide. Configure the new policy and URL, using the sign-in policies instructions in the *Juniper Networks Unified Access Control Administration Guide*.
4. Select the label you assigned to the templates, when you select the sign-in page from the drop down menu.
5. Click **Save Changes**.

To test your customized pages iteratively, you need to cycle through the preceding procedures to upload and associate a URL. You only need to download the templates once.

Understanding the Template Toolkit Language

This section includes an abbreviated description of the Template Toolkit language for Infranet Controller users. The section describes the most common directives and operators that designers may want to use when creating a customized page for the Infranet Controller. Although the Template Toolkit supports many powerful constructs, such as simple and complex datatypes, virtual methods, and interfaces to database systems, the custom sign-in pages use only a limited set of the Toolkit capabilities.

You can use the Template Toolkit for many applications that are far beyond the scope of this discussion. For complete information about the Template Toolkit, go to <http://www.template-toolkit.org>.

A Web page created with the Template Toolkit resembles a standard Web page. It can include HTML, XML, and JavaScript. Unlike a standard Web page, however, it can also include Template Toolkit directives, which you can use to add dynamic behavior to your pages.

A *directive* is a simple instruction that tells the template processor to perform an action and substitute the result into the document in place of the original directive. You can use directives for many purposes, such as iterating through a list of values (**FOREACH**), creating conditional statements (**IF/UNLESS/ELSE**), or including and processing another template file (**PROCESS/INCLUDE**).

When using directives in your code, note that:

- Directives are case-sensitive and all UPPER CASE.
- You must insert directives within the markup tags '<% ' and '%>'.



NOTE: The Template Toolkit documentation includes examples of using square bracket tags ([% %]) to mark directives. The Infranet Controller only supports angle bracket markup tags (<% %>).

- You can embed directives anywhere in a line of text.
- You can split directives across several lines.
- You can use the # character to indicate comments within a directive. The Template Toolkit language ignores characters after the # character. For more information on different types of comments, see “Template Comments” on page 15.
- The Template Toolkit language generally ignores insignificant white space within the directive.

In addition to directives, you can also use the Template Toolkit language to include loops, conditionals, variable substitutions, and other template files in your page.

The following sections describe common Template Toolkit tasks, directives, and operations you may want to use in your customized pages:

- “Accessing and Updating Variables and Files” on page 15
 - “GET Directive” on page 15
 - “SET Directive” on page 15
- “Creating Conditional Statements” on page 19
 - “Conditional Operators” on page 19
 - “IF and ELSIF Directives” on page 20
 - “SWITCH and CASE Directives” on page 20
- “Creating Looping Constructs” on page 21
- “Unsupported Directives” on page 21

For information about additional directives that you may use within the Template Toolkit language, such as **PROCESS** , **INCLUDE** , and **INSERT** , see the Template Toolkit documentation, available at <http://www.template-toolkit.org>.

Template Comments

You can include template comments by adding a **#** sign to the directive tag, as follows:

```
<%# This is a comment
      that comments out all
      of the lines that follow.
%>
```

You can include a single-line comment by leaving a space between the tag and the **#** sign, as follows:

```
<% # This is the first line comment, but the next line is not commented.
      IF foo...
      # This is another single-line comment.
%>
```

Accessing and Updating Variables and Files

GET Directive

The **GET** directive retrieves and outputs the value of the named variable.

```
<% GET foo %>
```

The **GET** keyword is optional. A variable can be specified in a directive tag by itself.

```
<% foo %>
```

SET Directive

The **SET** directive allows you to assign a value to existing variables or create new temporary variables.

```
<% SET title = 'Hello World' %>
```

The **SET** keyword is optional.

```
<% title = 'Hello World' %>
```

getText Directive

Used for multi-lingual support, the **getText** directive retrieves and outputs the localized value of the named variable.

```
<% getText foo %>
```

Table 4 on page 16 lists some of common variables used in the templates and their English values.

Table 4: English Values of Common Variables

Variable	English Definition
I18N_ACE_CONFIRM_PIN_COLON	Confirm PIN:
I18N_ACE_GEN_NEW_PIN	Generate New PIN?
I18N_ACE_GEN_PIN	Generate PIN
I18N_ACE_GEN_PIN_MESSAGE1	The system will now generate a PIN for you. Make sure that no one else can see your screen and then click Generate PIN to continue.
I18N_ACE_INVALID_PIN_FORMAT	Invalid PIN Format
I18N_ACE_INVALID_PIN_LENGTH	Invalid PIN Length
I18N_ACE_NEW_PIN_COLON	New PIN:
I18N_ACE_NEW_PIN_GENERATED	New PIN Generated
I18N_ACE_NEW_PIN_GENERATED_MESSAGE1	Your new PIN is
I18N_ACE_NEW_PIN_GENERATED_MESSAGE2	Be sure to remember it, because you need your PIN each time you sign in. When you have memorized it, click Continue to return to the sign-in page.
I18N_ACE_NEW_PIN_MESSAGE0	You must create a new Personal Identification Number (PIN) before you can sign in.
I18N_ACE_NEW_PIN_MESSAGE1	Your PIN should be
I18N_ACE_NEW_PIN_MESSAGE2	long.
I18N_ACE_NEW_PIN_MESSAGE3	Be sure to remember your PIN, because you need it to sign in.
I18N_ACE_NEW_PIN_MESSAGE4	If you prefer, the system can
I18N_ACE_NEW_PIN_MESSAGE5	generate a PIN
I18N_ACE_NEW_PIN_MESSAGE6	for you. Generated PINs are typically more secure.
I18N_ACE_NEW_PIN_MESSAGE7	If you decide not to create a new PIN now, click Cancel.
I18N_ACE_NEW_PIN_REQUIRED	New PIN Required
I18N_ACE_NEW_PIN_SIZE_MSG	Your PIN should be %1 to %2 digits long.

Table 4: English Values of Common Variables (*continued*)

Variable	English Definition
I18N_ACE_NEXT_TOKEN_MESSAGE1	Please enter an additional token code to continue.
I18N_ACE_NEXT_TOKEN_MESSAGE2	The server requires that you enter an additional token code to verify that your credentials are valid. To continue, wait for the token code to change and then enter the new code in the SecurID Token Code field.
I18N_ACE_NEXT_TOKEN_REQUIRED	Token Resync Required
I18N_ACE_PIN_MISMATCH	The Two PINs Entered Do Not Match
I18N_ACE_SAVE_PIN	Save PIN
I18N_ACE_TOKEN_CODE_COLON	SecurID Token Code:
I18N_CANCEL	Cancel
I18N_CLICK	Click
I18N_ENTER	Enter
I18N_HERE	Here
I18N_NETEACE_NEW_PIN_MESSAGE1	You must create a new Personal Identification Number (PIN) before you can sign in.
I18N_NETEACE_NEW_PIN_MESSAGE2	Wait for the token code to change, and then enter your current PIN* followed by the new token code in the SecurID Pass Code field. (If you want the system to generate a new PIN for you, select the System PIN checkbox.) Click Enter to advance to the new PIN assignment page. *If your administrator tells you that you no longer have a PIN, enter only the new token code in the Pass Code field.
I18N_NETEACE_NEW_SYSTEM_PIN	New System PIN Assignment
I18N_NETEACE_NEW_SYSTEM_PIN_MESSAGE1	The system will now generate a Personal Identification Number (PIN) for you.

Table 4: English Values of Common Variables (*continued*)

Variable	English Definition
I18N_NETEACE_NEW_SYSTEM_PIN_MESSAGE2	Wait for the token code to change, and then enter your current PIN* followed by the new token code in the SecurID Pass Code field. Make sure that no one else can see your screen and then click Generate PIN to continue. *If your administrator tells you that you no longer have a PIN, enter only the new token code in the Pass Code field.
I18N_NETEACE_NEW_USER_PIN	New PIN Assignment
I18N_NETEACE_NEW_USER_PIN_MESSAGE1	Create your new Personal Identification Number (PIN)
I18N_NETEACE_NEW_USER_PIN_MESSAGE2	Wait for the token code to change, and then enter your current PIN* followed by the new token code in the SecurID Pass Code field. Then, enter your new PIN. *If your administrator tells you that you no longer have a PIN, enter only the new token code in the Pass Code field.
I18N_NETEACE_NEXT_TOKEN_MESSAGE1	Please enter additional token codes to continue.
I18N_NETEACE_NEXT_TOKEN_MESSAGE2	The server requires that you enter additional token codes to verify that your credentials are valid. To continue, wait for the token code to change, and then enter your PIN followed by the new token code in the SecurID Pass Code field. Then, wait for the token code to change again and enter only the token code in the SecurID Token Code field.
I18N_NETEACE_NEXT_TOKEN_REQUIRED	Token Resync Required
I18N_NETEACE_PASS_CODE_COLON	SecurID Pass Code:
I18N_NETEACE_SYSTEM_PIN_REQUEST	System PIN
I18N_NETEACE_TOKEN_CODE_COLON	SecurID Token Code:
I18N_NEW_PIN_INVALID	New PIN entered is invalid. Please wait for the token to change and try login again.
I18N_TO_CONTINUE	to continue
I18N_USERNAME_COLON	Username:

If you want to customize a page to add your own message, edit the appropriate template file and enter your custom message using HTML tags. Make sure you do not change any of the surrounding logic. You can not edit the strings file(s) and change the value of the `I18N_*` variables. For example, suppose the template file has the following code:

```
<TABLE border="0" cellspacing="0" cellpadding="0">
  <TR>
    <TD><IMG border='0' src='/dana-na/imgs/logininfo.gif'
alt='Info' width='21' height='20'></TD>
    <TD><B><% GetText('I18N_ACE_NEW_PIN_REQUIRED') %></B></TD>
  </TR>
</TABLE>
```

To change the displayed message to “You must enter a new PIN number” instead of the default message “New PIN Required”, change the code as follows:

```
<TABLE border="0" cellspacing="0" cellpadding="0">
  <TR>
    <TD><IMG border='0' src='/dana-na/imgs/logininfo.gif'
alt='Info' width='21' height='20'></TD>
    <!--<TD><B><% GetText('I18N_ACE_NEW_PIN_GENERATED')
%></B></TD>-->
    <TD><B>You must enter a new PIN number</B></TD>
  </TR>
</TABLE>
```

Creating Conditional Statements

You can create conditional statements in the Template Toolkit language. Review the sample templates to find some example conditional statements. One of the simplest and most effective uses of conditional statements is in trapping error conditions.

For example, the following code snippet from the `LoginPage.thtml` template verifies the current user’s account status:

```
<% IF AccountDisabled %>
  <% LoginPageErrorMessage %>
<% ELSE %>
  <% IF LoginPageErrorMessage %>
    <% LoginPageErrorMessage %>
  <% END %>
<% END %>
```

Conditional Operators

You can use the following conditional operators:

Table 5: Conditional Operators

Operator	Description
<code>==</code>	is equal to
<code>!=</code>	is not equal to
<code><</code>	is less than
<code><=</code>	is equal to or less than

Table 5: Conditional Operators (*continued*)

Operator	Description
>	is greater than
>=	is equal to or greater than
&&	and
	or
!	not
and	and
or	or
not	not

IF and ELSIF Directives

You can use the **IF** and **ELSIF** directives to construct conditional behavior. Note that these are block directives, which means that you must use the **END** directive to indicate where each block ends. You may nest blocks indefinitely, provided you include an end statement for each nested block.

```
<% IF LoginPageErrorCode == 1002 %>
  <b> Your username or password is incorrect. Please reenter your credentials.
</b>
<%ELSIF LoginPageErrorCode == 1006 %>
  <b> The system is undergoing maintenance. Only administrators are allowed
to sign in at this time.</b>
<% END %>
```

SWITCH and CASE Directives

You may use the **SWITCH** and **CASE** directives to construct a multi-way conditional test. Note that these are block directives, which means that you must use the **END** directive to indicate where each block ends. You may nest blocks indefinitely, provided you include an end statement for each nested block.

```
<% SWITCH loginPageErrorCode %>
  <% CASE 1001 %>
    <b> Your session time has expired. </b>
  <% CASE 1006 %>
    <b> The system is undergoing maintenance. Only administrators are
permitted to sign in at this time. </b>
  <% CASE %> # default ...
<% END %>
```


Creating Looping Constructs

You may use directives such as **FOREACH** and **WHILE** to loop through blocks of code. Note that these are block directives, which means that you must use the **END** directive to indicate where each block ends. You may nest blocks indefinitely, provided you include an end statement for each nested block.

For example, the following sign-in page code loops through all of the authentication realms and displays them in a select list. The sample also uses the **RealmList** and **realm** predefined Infranet Controller values.

```
<select size="1" name="realm">
  <% FOREACH r = RealmList %>
    <option value="<% r %>" ><% r %> </option>
  <% END %>
</select>
```

Unsupported Directives

Juniper does not support the **USE**, **INTERPOLATE**, **TAGS**, **PERL**, or **RAWPERL** directives when creating Infranet Controller custom pages. Additionally, we do not recommend using the **CALL** or **FILTER** directives.

Getting Familiar with the Template Page Structure

The sample templates were created using the common method of designing with HTML tables. One of the easiest ways to see how the layout works is to set all of the table borders to a visible state. For example, change **<TABLE border="0">** to **<TABLE border="1">** wherever you find it in the **LoginPage.shtml** template.

If the Infranet Controller encounters errors during the login sequence, it displays errors in these two rows. The placement of the errors is strictly up to you, as long as the error directives occur within the **<form>** specification.

The table layout affords you a columnar layout. If you choose to maintain your customized design within the table layout, you may need to experiment with additions or deletions of table rows or cells. You can also change colors of cells, modify the locations of required fields, messages, or UI components.

You must always maintain the Template Toolkit directives that are included in each of the templates. These directives provide required information to the Infranet Controller and enable the Infranet Controller to communicate system state back to the end-user, appropriately. Deleting or modifying any of the provided directives can result in unexpected behavior or data corruption.

CGI Files

The Infranet Controller system generates a number of CGI files, which are called by the custom sign-in pages. Each file performs specific functions that are required by the Infranet Controller. You should not modify any of these calls nor should you attempt to create your own CGI files to take the place of the supplied CGIs. In particular, you may see a number of references in the custom sign-in pages to the three following CGI files:

- **login.cgi**—Handles authentication request between custom pages, the Infranet Controller, and any defined authentication servers.
- **welcome.cgi**—change, or new PIN generation.**welcome.cgi**—Handles a password change, or new PIN generation.
- **starter.cgi**—Verifies that client applications are running and allows the user to attempt another login, typically when too many users are concurrently signed in to the Infranet Controller.

Typically, in standard sign-in pages and sample custom pages the order of cgi usage is **welcome.cgi**, **login.cgi**, and then **welcome.cgi** again. For Host Checker pre-authorization, policies are evaluated in the first invocation of **welcome.cgi** and are passed to **login.cgi**. **login.cgi** takes the username and password parameters and determines the roles to assign. For Host Checker post-authorization, policies are evaluated only in the second invocation of **welcome.cgi**. When Host Checker and Cache Cleaner are configured, using the first URL to log in to the is not supported.



NOTE: When Host Checker is configured, using **login.cgi** as the first URL to log in to the Infranet Controller is not supported.

Using Templates from the samples.zip File

The **samples.zip** file contains the following templates which you can customize for use in your own environment. **Samples.zip** contains the most general set of templates. Some of the templates in **samples.zip** are also contained in **kiosk.zip** and **softid.zip**.

- Infranet Controller pre-authentication pages
 - “Logout.shtml” on page 69
 - “ExceededConcurrent.shtml” on page 48
 - “SSL.shtml” on page 107
 - “PleaseWait.shtml” on page 83
 - “SelectRole.shtml” on page 95
- ACE pre-authentication pages
 - “NewPin.shtml” on page 75
 - “NextToken.shtml” on page 76
 - “GeneratePin.shtml” on page 54
 - “ShowSystemPin.shtml” on page 103
- ACE with eTrust pre-authentication pages
 - “SM-NewPinSelect.shtml” on page 105
 - “SM-NewPinSystem.shtml” on page 105

- “SM-NewUserPin.shtml” on page 105
- “SM-NextToken.shtml” on page 106
- Password management pages
 - “Defender.shtml” on page 47
 - “GraceLoginUsed.shtml” on page 55
 - “PasswordChange.shtml” on page 80
 - “PasswordExpiration.shtml” on page 81

Customizing Look-and-Feel

You can customize the look-and-feel of your custom sign-in pages. This enables you to add your own corporate design, including logos, colors, and layout.

Required Components

You must include:

- Any template directives you find in the template files. This includes any code surrounded with template tags `<% %>`.
- Required HTML tags, such as the `<html>`, `<head>`, and `<body>` tags, although these may be consolidated into a separate template file which you can include using either the **PROCESS** or **INCLUDE** directives.
- Any `<form>` definitions.
- Any `<input>` statements.
- Any `<script>` statements and included scripts, typically Javascript.
- Any `<object>` statements.
- Any `<embed>` statements.
- Any `onload`, `onUnload`, or other event triggers in the `<body>` tag.
- Any of the required variables, directives, or other items as noted in the commented section at the beginning of each template file.

You can exclude:

- Any HTML presentation code that is not a template directive, such as table tags, images, or style tags.

Displaying Custom Logos

You can change the templates to display your company's logo by following these steps:

1. Open the template in a text editor or HTML editor.
2. Search for either of the two strings:

```
src="welcome.cgi?p=logo&signinId=<%signinId%>"
```

```
src="welcome.cgi?=logo"
```

3. Replace the string with the following:

```
src="<% Home %>/imgs/custom-logo.gif"
```

where *custom-logo.gif* is the new logo to be displayed. *custom-logo.gif* must be uploaded to the Infranet Controller as part of your zip file. For more information, see "Downloading Sample Templates and Uploading Files to the Infranet Controller" on page 12.



NOTE: For *Remediate.html*, use `<%signinId%>` instead. For example:

```
src="<% signinID %>/imgs/custom-logo.gif"
```

Simplifying the Code in the Sample Templates

One of the first things you can do is simplify the code in the templates, by using the Template Toolkit **PROCESS** directive to extract common code into an **include** file. By moving common code to separate files, you can eliminate a certain amount of redundant code, thereby decreasing the size of the template files, and improving the readability of the files.

Let's take a look at one of the template files, **LoginPage.html**. Almost every designer who wants to customize the custom sign-in pages will likely want to customize **LoginPage.html**, which is the first screen that end-users see. If you have not downloaded the **Samples.zip** file, do so now, following the instructions in "Downloading Sample Templates and Uploading Files to the Infranet Controller" on page 12.

Open the **LoginPage.html** file in a text editor or HTML editor.

Version Directive

Each template file starts with a version directive, similar to the following:

```
<%# NetScreen Page Version 1001 %>
```

Do not modify this directive.

Comment Section

Each template file then contains a set of comments that describe template variables, JavaScript, and HTML code required in that particular template. These comments can help you understand what directives and other elements of the file you must maintain, and give hints about other directives, variables, and methods that you can use to further customize the template.



NOTE: When you have completed your customization, you can consider removing the comment section from your files. The removal of this section decreases the size of the template files, and can possibly speed up page download times.

Header Tags

With the exception of the **Remediate.thtml** file, all of the template files contain the following HTML tags:

```
<html>
<head>
  <meta http-equiv="Content-Language">
  <meta http-equiv="Content-Type" content="text/html">
  <meta name="robots" content="none">
```

You can extract these tags into a separate **.thtml** file. If you ever need to change any of these tags, for example, to modify the meta tag definitions, you can then modify one file and the changes are propagated throughout all the files.

To create a simple header template file:

1. Open a new text file in your favorite text editor or HTML editor.
2. Cut and paste the preceding header lines into the file. You can cut these from one of the template files.
3. Save the file into the directory where the other template files are saved, naming the file **customhead.thtml**.



NOTE: Do not name the file **header.thtml**. Along with **config.thtml** and **footer.thtml**, this name may already be used by the Infranet Controller system software. You can call the files whatever you want, as long as they do not use these reserved names. They must end in the **.thtml** file extension.

4. Open each template file and delete the header lines.
5. In the place of the header lines, insert the following directive:

```
<% PROCESS customhead.thtml %>
```

The command **PROCESS** must be in all UPPER CASE. If you name the header file something other than **customhead.thtml**, then use that name in the directive instead.

6. When you zip your template files up, make sure you include the **customhead.thtml** file.

Copyright and Footer Tags

You might want to perform a similar task with the footer tags. This topic shows how to create a copyright notice in a separate include file.

1. Open a new text file in your favorite text editor or HTML editor.
2. Copy the following directive into the file:

```
<% year = 2005;
```

```
company = "Juniper Networks";  
copyr = "Copyright $year $company" -%>
```

You can change the company name to your own company name.

3. Save the file into the directory where the other template files are saved, naming the file **customconfig.thtml**.
4. Open another new text file in your favorite text editor or HTML editor.
5. Copy the following directive and HTML to the file:

```
<div id="footer">  
  &copy; <% copyr -%>  
</div>  
</body>  
</html>
```

6. Save the file into the directory where the other template files are saved, naming the file **customfoot.thtml**.
7. In your **LoginPage.thtml** file, add the following directive just prior to the **PROCESS** directive for **customhead.thtml**. The lines should look like:

```
<% PROCESS customconfig.thtml %>  
<% PROCESS customhead.thtml %>
```

8. In your **LoginPage.thtml** file, scroll to the end of the file.
9. Replace the last two lines that contain the closing **</body>** and closing **</html>** tags with the following directive:

```
<% PROCESS customfoot.thtml %>
```

10. Save the **LoginPage.thtml** file.

Page Banner

You can also create a common banner for your pages, eliminating the repetitious use of images and HTML code in every file. The following sample code creates a banner using an H1 header for the text displayed in the banner. Note the commented image tag, which you could use instead of the H1, if you prefer to use an image, for example, a logo.

1. Open a new text file in your favorite text editor or HTML editor.
2. Copy the following HTML into the file:

```
<div id="banner">  
  <h1 class="banner">Sample</h1>  
  <!--   
</div>
```

3. Save the file into the directory where the other template files are saved, naming the file **custombanner.thtml**.

4. In your **LoginPage.thtml** file, add the following directive just after the **<body>** tag.

```
<body id=" body" onload="FinishLoad()">
<% PROCESS custombanner.thtml %>
```

5. Save the **LoginPage.thtml** file.

Processing LoginPage.thtml

Now you can process the template to see the results of making the changes. Keep in mind the following requirements:

- You must have successfully installed and configured the Template Toolkit on your local system.
- All of the template files, including the custom header, custom footer, and custom config files must be saved into the same directory along with **LoginPage.thtml**.
- If you intend to upload your files to the Infranet Controller, instead of processing them locally first, you can ignore this instruction. You must begin each file with the directive **[% TAGS <% %> %]** so that you can use angle brackets on any directives you add to the templates. By default, the Template Toolkit recognizes square brackets (**[]**) instead of angle brackets (**<>**) as delimiters for directives. The Infranet Controller recognizes angle brackets only. Adding the **TAGS** directive tells the local Template Toolkit utilities that you are using angle brackets.

To process **LoginPage.thtml**:

1. Open a command window.
2. Navigate to the directory in which you have saved the **LoginPage.thtml** file, **customhead.thtml**, **customconfig.thtml**, and **customfoot.thtml**. For example, **/Templates/Sample/**.
3. Enter the following command:

```
tpage LoginPage.thtml > index.html
```

If you get no errors and the system returns a blank directory prompt, the process completed successfully.

4. Open a browser window.
5. Select **File > Open** and navigate to the directory where you have saved **LoginPage.thtml**.
6. Select **index.html**.

You should be able to see a copyright statement at the bottom of the page. The directive in **customfoot.thtml** pulled the data from **customconfig.thtml** and substituted the year and company name into the substitution variables **\$date** and **\$company**. Then, during the processing of **LoginPage.thtml**, **customfoot.thtml** was included based on the use of the **PROCESS** directive.



NOTE: Two warnings:

- If you received errors or if you were unable to run `tpage`, you need to check your Template Toolkit installation and documentation. You may not have the proper environment variable set to point to the Template Toolkit installation location or to your Perl distribution.
- Once you create a file, such as `index.html` in the preceding example, you must delete or move the file if you intend to create another version of the same file. The Template Toolkit `tpage` utility does not overwrite a file of the same name.

Changing Colors in the Table-Based Layout

Changing colors in the table-based layout is a very simple operation. All you need to do is replace color codes in the templates with the color codes representing the colors you prefer. Color codes are expressed in the hexadecimal format (`#000000`). If you want to change everything that is white to black, replace all occurrences of `#FFFFFF` to `#000000`.

In the **UI Options** page of a given role, you can also change banner, background, and link colors.

Changing the Layout

One of the simplest ways to customize the presentation of the pages is to modify the banner, colors, and table format. You can add table rows (`<tr>`) and table cells (`<td>`), graphics, change the `custom-logo.gif` to your own company logo, and change the background color of one or more table cells. You can also move pre-configured objects, such as the username field, password field, realm drop down menu, and submit button to other locations within the table. We do not go into detail about how to accomplish this. You should be familiar with HTML tables and be prepared to go through a trial and error process.

If you prefer to separate the presentation from the content, consider using cascading style sheets (CSS).

Using Cascading Style Sheets

One of the most dramatic changes you can make to the presentation style of the templates is to convert them fully to use cascading style sheets (CSS). In some templates you can already find brief examples of CSS codes, such as the use of `<DIV>` and `<SPAN>` tags. In those cases, it is best to leave those tags in place. You can, however, replace much of the HTML code around those tags, as well as around Template Toolkit directives.

As mentioned before, the sample templates use a traditional table-based design model. In effect, the design uses HTML tables as containers for all other objects on the page. Table rows and cells contain all graphics, error messages, fields, and buttons.

Although tables can simplify some elements of HTML design, they can complicate others, not to mention that designing with tables means that you must use tables on every page, if you intend to maintain the same basic layout. This adds complexity and decreases

download times, as the end-user's browser must parse each table in each page every time it loads a page.

CSS provides an alternative. CSS consists of a set of styles that can be embedded in each HTML page or that can be stored in a separate `.css` file and referenced in the HTML files with a `<link>` tag. This topic is not meant to be a comprehensive CSS tutorial. It assumes you already have some familiarity with CSS. If you do not, you can find references on the Web, starting with the CSS specification at <http://www.w3c.org>.

For the purposes of this discussion, we will describe how to use CSS to eliminate redundant items, presentation information, and tables from your design. By doing so, you can improve the readability of your template files, and can consolidate almost all presentation instructions into one file.

Embedding CSS

You can embed CSS styles into an individual HTML page. If you plan on customizing only one file, you might find this approach easiest. Otherwise, you gain the most benefit by using a standalone CSS file and linking it into your HTML files.

Attaching a Standalone CSS

For the purposes of this discussion, we will introduce a simple CSS that will serve as an example method for replacing tables as a design element.

You can copy the following CSS into a text file and save it into the same directory with `LoginPage.html`. Save it as `sample.css`.

```
body { // Configures body margins
    margin:10px 10px 0px 10px;
    padding:0px;
}

span.error ( // Formats errors with bold and background
    font-weight: bold;
    background: silver;
}

#leftcontent { // Styles left column
    position: absolute;
    left:10px;
    top:69px;
    width:199px;
    background:#809FB0;
    border:1px solid #000000;
    text-align:center
}

#centercontent { // Styles center column
    background:#EFEFC9;
    margin-left: 175px;
    margin-right:11px;
    border:1px solid #000000;
    voice-family: "\"}\"\"";
    voice-family: inherit;
```

```
margin-left: 199px;
margin-right: 0px;

#banner { // Styles the top banner
background:#C0C0B0;
height:60px;
border-top:1px solid #000000;
border-left:1px solid #000000;
border-right:1px solid #000000;
voice-family: "\"}\"\"";
voice-family: inherit;
height:59px;
}

p,h1,h2,h3,pre { // Sets common properties of these
margin:0px 10px 10px 10px;// paragraph elements
font-family:Arial, Verdana, Helvetica, sans-serif;
}

p{ // Styles paragraph element
font-size:12px;
text-align:left
}

h1,h2,h3 ( // Sets common elements for
padding-top:10px; // headings.
font-weight:bold;
}

h1 { // Sets size of H1 heading
font-size:18px;
}

h2 { // Sets size of H2 heading
font-size:14px;
}

h3{ // Sets size of H3 heading
font-size:12px;
}

#banner h1 { // Sets heading selector for banner
font-size:36px;
padding:10px 10px 0px 10px;
margin:0px;
font-weight: bolder
}

#footer { // Sets footer selector
border-top:3px solid #000000;
text-align:center;
font-family:sans-serif;
font-size:10px
}

a { // Sets common styles for anchor
font-family:sans-serif;
```

```

    font-size:14px;
    font-weight:bold;
}

a:link { // Sets link style
    text-decoration: underline;
    text-align:right;
    color: black;
}

a:hover { // Sets hover style for links
    color:white;
}

```

To make the connection between the CSS stylesheet and the page, you must include a `<link>` tag prior to the closing head tag `</head>`, as follows:

```

<link rel="stylesheet" type="text/css" href="sample.css" />
</head>

```

If you store the stylesheet in a directory other than the root directory where you keep your template files, make sure you express the path correctly in the href attribute of the link tag. For example, if you keep all of your stylesheets in a directory one level above the template file, you might indicate it as follows:

```

<link rel="stylesheet" type="text/css" href="../../sample.css" />
or

```

```

<link rel="stylesheet" type="text/css" href="css/sample.css" />

```

This CSS defines the presentation style of all of the major components of the page. Additionally, if you want to change the look-and-feel of the page, while keeping the basic structure intact, you can create another CSS that uses the same style names, but that modifies the definitions. Without making any changes to your template file, you can apply an entirely unique style.

Here is a copy of the **LoginPage.thtml** file (excluding the comments section) after it has been converted to a CSS-based page. Compare it with the template file that you download from the Infranet Controller. You might notice that virtually all of the presentation codes, things such as color settings, font settings, and border settings are missing from the following code. All of the presentation code has been extracted to the CSS stylesheet.

```

<%# NetScreen Page Version 1001 %>

```

```

<!-- The following line, when uncommented, allows you to process
the .thtml file locally using the tpage utility, without having to
replace the angle brackets on all of the Infranet Controller template
directives.

```

```

The TAGS directive tells the Template Toolkit utilities to substitute
angle brackets for the default square brackets. This line should be
on the first line of the file when you are testing locally. Prior to
uploading to the Infranet Controller, move and comment the line, or remove it
from the file completely.

```

```
-->
<!--[% TAGS <% %> %]-->

<% PROCESS customconfig.shtml %>

<% PROCESS customhead.shtml %>

<title>Instant Virtual Extranet</title>

<link rel="stylesheet" type="text/css" href="sample_two.css" />

<script>
<!--
    JavaScript removed for readability. See template file for actual JavaScript.
//-->
</script>
</head>
<body id="body" onload="FinishLoad()">

<% PROCESS custombanner.shtml %>

<!-- ////////// -->
<!-- BEGIN LEFT COLUMN -->
<!-- ////////// -->

<div id="leftcontent">

<!-- ////////////////////////////////// -->
<!-- BEGIN CUSTOM CONTENT LEFT COLUMN (Optional) -->
<!-- ////////////////////////////////// -->

<!-- Add more content here, if needed. -->

<!-- ////////////////////////////////// -->
<!-- END CUSTOM CONTENT LEFT COLUMN -->
<!-- ////////////////////////////////// -->

<!-- ////////////////////////////////// -->
<!-- BEGIN AUTHENTICATION TEMPLATE DIRECTIVES -->
<!-- and FORM USERNAME/PASSWORD/REALM PROMPTS -->
<!-- DO NOT DELETE. Must be included in page. -->
<!-- ////////////////////////////////// -->

<form name="frmLogin" action="login.cgi" method="post" autocomplete="off"
onsubmit="return Login()">

<input type="hidden" name="tz_offset">
<%IF !AnonymousAuthentication && !CertificateAuthentication%>
    <% FOREACH prompt = prompts %>
        <%NEXT IF !prompt.required %><br>
        <p><% prompt.promptText %></p>
<input type="<% prompt.type %>" name="<% prompt.name %>" size="20">
<% END %>

<% IF RealmList.size == 0 %>
    <p>LoginRealm</p>
    <input type="text" name="realm" size="20">
    <% ELIF RealmList.size == 1 %>
```

```

        <input type="hidden" name="realm" value="<% RealmList.0 %>">
    <% ELSE %>
    <p>LoginRealm</p>
    <select size="1" name="realm">
    <% FOREACH r = RealmList %>
        <option value="<% r %>" ><% r %></option>
    <% END %>
    </select>
    <% END %>
<%ELSE%>
    <input type="hidden" name="realm" value="<% RealmList.0 %>">
<%END%>
    <input type="submit" value="Sign In" name="btnSubmit">
</form>

<!-- ////////////////////////////////// -->
<!-- BEGIN CUSTOM CONTENT LEFT COLUMN -->
<!-- ////////////////////////////////// -->

<p>You are not yet authorized!</p>

<!-- ////////////////////////////////// -->
<!-- END CUSTOM CONTENT LEFT COLUMN -->
<!-- ////////////////////////////////// -->

</div>

<!-- ////////////////////////////////// -->
<!-- END LEFT COLUMN -->
<!-- ////////////////////////////////// -->

<!-- ////////////////////////////////// -->
<!-- BEGIN CENTER COLUMN -->
<!-- ////////////////////////////////// -->

<div id="centercontent">

<!-- ////////////////////////////////// -->
<!-- BEGIN PASSWORD MANAGEMENT TEMPLATE DIRECTIVES -->
<!-- DO NOT DELETE. Must be included in page. -->
<!-- Added for Account Disabled Case for Password Managemnt -->
<!-- ////////////////////////////////// -->

<% IF AccountDisabled %>

<!-- ////////////////////////////////// -->
<!-- Account Disabled Case -->
<!-- ////////////////////////////////// -->

<span class="error"><% LoginPageErrorMessage %></span>

<% ELSE %>
    <% IF LoginPageErrorMessage %>
        <span class="error"><% LoginPageErrorMessage %></span>
    <% END %>
<% END %>

<!-- ////////////////////////////////// -->

```

```
<!-- END PASSWORD MANAGEMENT DIRECTIVES -->
<!-- ////////////////////////////////// -->

<!-- ////////////////////////////////// -->
<!-- Optional error handling codes -->
<!-- ////////////////////////////////// -->

<span class="error">
<% IF PageErrorCode == 1020 %>
    You have successfully changed your password.
<% END %>
</span>

<span class="error">
<% IF PageErrorCode == 1021 %>
    Account Disabled. Please contact Administrator for help.
<% END %>
</span>

<span class="error">
<% IF PageErrorCode == 1023 %>
    Account Expired. Please contact Administrator for help.
<% END %>
</span>

<!-- ////////////////////////////////// -->
<!-- END optional error handling codes -->
<!-- ////////////////////////////////// -->

<!-- ////////////////////////////////// -->
<!-- BEGIN CUSTOM CONTENT CENTER COLUMN -->
<!-- Add any content below that you want to appear in the center column. -->
<!-- You can add a mix of headings, paragraphs, graphics or any other HTML.-->
<!--//////////////////////////////////// -->

<h1>Welcome to XYZ Company!</h1>

<h2>New Employee Benefits!</h2>
<p>Per munere latine officiis ad. Quo te elit vivendum, mea aperiri integre
periculis ex, ea sea corpora consecetuer. </p>
<h2>Company Party!</h2>
<p>Dolore efficiantur vim ea, alia putant vivendo at pri. Odio cetero tibi que no vis.
Mea vero delenit ad. </p>
<h2>Corporate Travel Advisory</h2>
<p>Cu zzril expetenda has, ludus utinam instructor nam an, id eam fugit putent
possit. Mazim epicuri contentiones mel ut, pri atqui maiestatis ei. </p>
<p>Tempor oportere usu ne. Ea aperiam sanctus oportere vel.</p>

<!-- ////////////////////////////////// -->
<!-- END CUSTOM CONTENT CENTER COLUMN -->
<!-- ////////////////////////////////// -->

</div>

<!-- ////////////////////////////////// -->
```

```
<!-- END CENTER COLUMN -->
<!-- ////////////////////////////////// -->

<% PROCESS customfoot.thtml %>
```

Customizing Error Handling and Navigation

The Template Toolkit provides a rudimentary level of error handling capability. In most cases, errors are handled sufficiently by the included logic. However, you might want to modify the behavior of the pages, given a particular error. For example, you might want to redirect a user to your own specific custom error page on an “Account Disabled” error.

You can redirect based on specific individual errors or based on any error occurring at all. The following samples illustrate both methods.

Redirecting Based on Any and All Errors

If you want to redirect to your own Web page based on the occurrence of any and all errors, you can implement logic, as follows:

```
<head>
  <% IF LoginPageErrorMessage %>
    <meta http-equiv="refresh" content="0;url=DisplayErrorPage.html" >
    <title>Redirect</title>
  </head>
```

Redirecting Based on Specific Errors

If you want to redirect to your own Web page based on the occurrence of a specific error, such as the “Account Disabled” error, you can implement logic, as follows:

```
<head>
  <% IF LoginPageErrorCode == 1021%>
    <meta http-equiv="refresh" content="0;url=AcctDisabledError.html" >
  <% ELSIF PageErrorCode == 1023 %>
    <meta http-equiv="refresh" content="0;url=AcctExpiredError.html" >
  <% END %>
  <title>Redirect</title>
</head>
```

For more information on error codes, see “LoginPageErrorCode” on page 66.

Defining Error Message Locations

You can locate error messages wherever you want on the page. Keep in mind, however, that you cannot move the error handling code relative to its position within the `<form/>` tags. For example, do not reverse the positions of two directives appearing within the body of `<form/>` tags or move the error handling code outside of the `<form/>` tags.

Customizing Error Messages

You can customize some of the error and confirmation messages your end-users might see. The Infranet Controller produces some error messages that you cannot change. Table 7 on page 66 lists those that you can change.

Although you cannot change any messages on the Infranet Controller itself, you can customize error messages that will appear instead of the messages originating on the Infranet Controller. Using the Template Toolkit **IF** directive, you can test for a message based on an error's unique error code, and expressed by the predefined template variable **PageErrorCode**. In the directive block, you include the customized message, as shown in the following example:

```
<% IF PageErrorCode == 1020 %>
    You have successfully changed your password.
<% END %>

<% IF PageErrorCode == 1021 %>
    Account Disabled. Please contact administrator at X5540 for help.
<% END %>

<% IF PageErrorCode == 1023 %>
    Account Expired. Please contact administrator at X5541 for help.
<% END %>
```

You might include the preceding code in your **LoginPage.thtml** file following the form (**<form></form>**) that prompts the user for login credentials. You can see that the example messages in the preceding sample include phone extension numbers, to help end-users identify the correct administrator to call after encountering specific problems.

Localizing Custom Sign-In Pages

By default, text strings are in English. Administrators supporting non-English users may need to configure the sign-in pages to provide localized text labels. This can only be done on a per-sign-in page basis. For multi-language support, Administrators must configure different sign-in pages for different locales. For further customizations, you can upload customized sign-in pages using the Template Toolkit. Contact Juniper Networks Technical Support for details.

You can create localized custom sign-in pages, using the downloadable sample template files. You can download the samples from **Signing In > Sign-in pages > Upload Custom Pages > Upload Custom Sign-In Pages** page.

You include the localized version of the template files in separate directories, one directory per language, in your uploaded zip file. The Infranet Controller recognizes all of the **.thtml** files in the root directory of the zip file as default pages. When creating the subdirectory for a set of given language files, name the subdirectory with the Accept-Language header 2-char abbreviation. Retain the default names for the localized file names.

At runtime, the Infranet Controller notes the default language of the browser, then searches the subdirectory corresponding to the 2-char **Accept-Language** abbreviation to find the specific **.thtml** file. If the Infranet Controller locates the file, the Infranet Controller uses that **.thtml** file. If the Infranet Controller cannot locate the file, the Infranet Controller uses the default page in the root directory.



NOTE: Use the Accept-Language abbreviation **zh-cn** if you plan to use either of the abbreviations **zh-cn** or **zh-sg**.

The directories should contain full sets of template files, as required to implement custom sign-in pages.



NOTE: For a list of acceptable Accept-Language header abbreviations, see <http://www.oasis-open.org/cover/iso639a.html>.

Creating a Localized Set of Custom Sign-In Pages

To create a localized set of custom sign-in pages, perform the following procedure.

1. Select **Signing In > Sign in > Sign in pages**.
2. Click **Upload Custom Pages**.
3. Click the link for the sample templates you want to localize.
4. Once downloaded, unzip the files.
5. Create a subdirectory in the sample templates directory and name it with the 2-character identifier that specifies the language into which you intend to translate the template files. For example, if you are translating into French, name the directory **fr**. Create subdirectories for each language into which you want to translate the templates, and copy all template files and the images directory into each subdirectory.
6. Copy all of the templates and the images directory (**/imgs**) and paste them into the new folder.
7. Translate the template files.
8. Change the paths in the translated files to point to the correct path. For example, you need to update the paths to images in the **/imgs** subdirectory, otherwise, they will continue to point to the default directory.
9. When translation is completed, zip the files and upload to the Infranet Controller.
10. To test the languages, make sure you update the language option in your Web browser, first.

Upgrade Considerations

Consider the following when upgrading to the latest version of the custom sign-in pages:

- You must update the version number in your templates to match the current version. If you do not update your version number, the default page appears instead of your custom page. Alternatively, you can copy your custom content to the new templates.
- Old variable names may change and new variables may be added. It is recommended that you convert old variable names to their new counterparts as the default values for the old variables may no longer exist. If you do not want to update your variable

names, you can select the **Skip validation checks during upload** option in the **Upload Custom Sign-In Pages** page. If you select this option, you should review all your custom pages to ensure that they are still functioning correctly.

- Some formatting, such as text strings and color, change from prior versions. Review your page to ensure that the changes are acceptable.

Custom Templates Reference

The following topics describe each of the template files, JavaScript, template variables, some form variables, their uses, functionality, and customizability. The topics are organized alphabetically. Within the reference, you can find definitions for:

- Infranet Controller pre-authentication pages

You may customize a variety of standard pre-authentication pages, including the standard sign-in page (out page (and Cache Cleaner start page ((You may customize a variety of standard Infranet Controller pre-authentication pages, including the standard sign-in page (**LoginPage.thtml**), the failed authentication page (**SSL.thtml**), the sign out page (**Logout.thtml**), the sign-in warning page (**ExceededConcurrent.thtml**), the Host Checker start page (**PleaseWait.thtml**), and the select from multiple roles page (**SelectRole.thtml**).

- ACE pre-authentication pages

The **SoftID.zip** file enables you to customize Infranet Controller pages for use with the RSA Soft ID client.

When the end-user browses to the URL of the SoftID custom sign-in page, the page prompts the user to enter a PIN by way of the SecurID application on his local machine. If the application accepts the PIN, the end-user is logged in.

To configure the SoftID custom sign-in page, download the **Samples.zip** and **SoftID.zip** files. Unzip Samples.zip to a directory first, and then unzip the **SoftID.zip** file to the same location, overwriting any duplicates. Once you have customized the files to your liking, zip the entire set of files and upload them to the Infranet Controller.

To ensure that the New Pin and Next Token pages are customized for SoftID authentication, copy **NewPin.thtml** to **GeneratePin.thtml** in the **SoftID.zip** file and upload the modified zip file to the Infranet Controller for the custom sign-in page.

- ACE with eTrust pre-authentication pages

The ACE with eTrust pre-authentication pages are named with the SM prefix, for SiteMinder.

- Password management pages
- These include pages such as the secondary authentication server login page (Template variables
- Kiosk pages

The **Kiosk.zip** file contains templates that enable you to customize Infranet Controller pages for use by Kiosk users. This sample shows how you can implement a custom UI

to circumvent keystroke loggers. When you run the sample **LoginPage.thtml** (after it has been uploaded to the Infranet Controller) you see a Virtual Keyboard. The password field is disabled, so an end-user must use the Virtual Keyboard to enter a password. Because the normal keyboard is disabled, a keystroke logger cannot capture the password.

Meeting pagesThe authentication pages used by Secure Meeting attendees.



NOTE: When uploading templates to the Infranet Controller, you must include all of the THTML pages included in the original zip files for the Infranet Controller to accept the upload.

- JavaScript

Many of the templates include optional JavaScript functions that you can use to perform a variety of tasks on that particular page. For example, the header in **LoginPage.thtml** contains several JavaScript functions. Most of these functions handle cases where you associate multiple authentication realms with a single sign-in URL. You can also add your own JavaScript functions to perform presentation effects, or to modify the default navigation by redirecting to your own Web pages, in some cases.

<failedPolicy>

	Tag used to go through list of failed policies.
Object Type	Template variable
Included In	Remediate.thtml
Usage	Required. Do not modify.
Example	None
Related Topics	"Remediate.thtml" on page 89

<failedPolicy>.name

	Name of the policy that fails during remediation.
Object Type	Template variable
Included In	Remediate.thtml
Usage	Required. Do not modify.
Example	None

Related Topics “Remediate.thtml” on page 89

<failedPolicy>.remediation

	Custom instructions as defined in the remediation policy.
Object Type	Template variable
Included In	Remediate.thtml
Usage	Optional
Example	None
Related Topics	“Remediate.thtml” on page 89

action

	An HTML form element that provides the action that is meant to occur, usually on a submit button click or POST .
Object Type	HTML form element
Included In	All templates with forms
Usage	Required
Example	The following fragment from a <form> field runs the login.cgi Perl script on the Infranet Controller: <code>action="login.cgi"</code>
Related Topics	“autocomplete” on page 43

AgentInstall.thtml

	Displays the “Launching the UAC Agent” page after a user installs the Odyssey Access Client.
Object Type	Template
Usage	Optional. This page appears after a user starts the Odyssey Access Client installation process and detects if ActiveX or Java is enabled.
Required Variables	“DSLlaunchURL” on page 48
Example	None

Related Topics “PleaseWait.shtml” on page 83

AnonymousAuthentication

Use this variable to specify that users signing into an anonymous realm should not see the username and password fields in the sign-in page.

Object Type Template variable

Included In LoginPage.shtml,

Usage Optional. The Infranet Controller sets this value to true if the authentication realm is set to an anonymous server. If you use this variable, the Infranet Controller only displays the sign-in page to users when they encounter one of the errors described in “LoginPageErrorCode” on page 66. If you set **AnonymousAuthentication**, the realm is available in **RealmList.0**. You need to send the realm in the form as a hidden variable.

Example The following sample checks to see if **AnonymousAuthentication** and **CertificateAuthentication** are set to false. If the variables are set to false, the following directives generate authentication fields, setting the size of the input fields to 20 characters.

```
<%IF !AnonymousAuthentication && !CertificateAuthentication%>
    <% FOREACH prompt = prompts %>
        <%NEXT IF !prompt.required %>
        <% prompt.promptText %>
        <input type="<% prompt.type %>" name="<% prompt.name %>" size="20">
    <% END %>
```

Related Topics

- “CertificateAuthentication” on page 45
- “RealmList” on page 88

authenticate()

Authenticates a user.

Object Type JavaScript function

Included In GeneratePin.shtml (ACE/SoftID), LoginPage.shtml (ACE/SoftID), NewPin.shtml (ACE/SoftID), NextToken.shtml (ACE/SoftID)

Usage Required

Example The following examples illustrate the use of the **authenticate()** function.

From ACE/SoftID **LoginPage.shtml**. This function checks to see whether or not the user is authenticated. If the user is not authenticated, the function displays an error from the

authentication server. Otherwise, it passes the authentication credentials from the auth server to the **frmLogin** HTML form in **LoginPage.shtml**:

```
function authenticate()
{
    var rc = -1;
    var time = <% softid_time %> ;
    var user = document.frmLogin.username.value;
    var state = 0;

    if(typeof(document.sdui.sdAuth) == "undefined")
    {
        rc = -1;
    }
    else
    {
        if (error != "")
        {
            alert(error);
        }
        rc = document.sdui.sdAuth(time, user, state);
    }
    if (rc == 1)
    {
        document.frmLogin.username.value = document.sdui.getUsername();
        document.frmLogin.password.value = document.sdui.getPasscode();
        document.frmLogin.submit();
    }
    else
    {
        document.cancelForm.submit();
    }
}
```

From ACE/SoftID **GeneratePin.shtml**. The following function enables the user to generate a new PIN for signing in to an authentication server:

```
function authenticate()
{
    var min = parseInt(document.minmaxForm.minlen.value);
    var max = parseInt(document.minmaxForm.maxlen.value);
    var alp = parseInt(document.minmaxForm.alphanumeric.value);
    var sys = document.minmaxForm.systempin.value;

    if(typeof(document.sdui.sdPin) == "undefined")
    {
        rc = -1;
    }
    else
    {
        if (error != "")
        {
            alert(error);
        }

        rc = document.sdui.sdPin(min, max, <% secid_pinselectmode %>,
alp, sys);
    }
}
```

```

if (rc == 1)
{
    document.frmNewPin.password.value = document.sdui.getPin();
    document.frmNewPin.password2.value =
        document.frmNewPin.password.value;
    document.frmNewPin.secidaction.value = "Save PIN";
    document.frmNewPin.secidactionSavePin.value = "Save PIN";
    document.frmNewPin.submit();
}
else if(rc == 2)
{
    document.frmNewPin.password.value = "";
    document.frmNewPin.password2.value = "";
    document.frmNewPin.secidactionCancel.value = "Cancel";
    document.frmNewPin.submit();
}
else
{
    document.cancelForm.submit();
}
}

```

- Related Topics**
- "GeneratePin.shtml" on page 54
 - "LoginPage.shtml" on page 65

autocomplete

Prevents the form from auto-filling username or authentication realm field entries using cached values.

Object Type HTML form element

Included In All templates with forms

Usage Optional

Example

```

<form name="frmLogin" action="login.cgi" method="post" autocomplete="off"
onsubmit="return Login()">

```

Related Topics None

Cancel.shtml

Displays a message to the user informing him that the sign-in request has been cancelled.

Object Type Template

Included In Usage Required
Included In Included In

Required Variables None

Example None

Related Topics None

ccInAcTimeout

Specifies the Cache Cleaner login inactivity in minutes.

Object Type Template variable

Included In LoginPage.thtml

Usage Required

Example The following code sets the timeout interval to 60000 milliseconds times the smaller value of `ccInAcTimeout` or `hcInAcTimeout`.

```
<% IF hcInAcTimeout > ccInAcTimeout %>
    window.setTimeout("deletepreauth()", 60000 * <% ccInAcTimeout %>);
<% ELSE %>
    window.setTimeout("deletepreauth()", 60000 * <% hcInAcTimeout %>);
<% END %>
```

Related Topics "LoginPage.thtml" on page 65

ccRunning

Specifies whether or not Cache Cleaner is currently running.

Object Type Template variable

Included In LoginPage.thtml

Usage Required

Example The following code sets the timeout interval based on whether Host Checker and Cache Cleaner are both running or only Host Checker.

```
<% IF hcRunning && ccRunning %>
    <% IF hcInAcTimeout > ccInAcTimeout %>
        window.setTimeout("deletepreauth()", 60000 * <% ccInAcTimeout %>);
    <% ELSE %>
        window.setTimeout("deletepreauth()", 60000 * <% hcInAcTimeout %>);
    <% END %>
<% ELSIF hcRunning %>
    window.setTimeout("deletepreauth()", 60000 * <% hcInAcTimeout %>);
<% END %>
```

Related Topics "LoginPage.thtml" on page 65

CertificateAuthentication

Use this variable if you want to specify that users signing into a certificate realm should not see the username and password fields in the sign-in page.

Object Type Template variable

Included In LoginPage.thtml,

Usage The Infranet Controller sets this value to true if the authentication realm is set to a certificate server. If you use this variable, the Infranet Controller only displays the sign-in page to users when they encounter one of the errors described in “LoginPageErrorCode” on page 66. (For information about hiding this page from users authenticated against a eTrust SiteMinder server using client-side certificate authentication, see “Login()” on page 64.) If you set **CertificateAuthentication**, the realm is available in **RealmList.0**. You need to send the realm in the form as a hidden variable.

Example The following code tests for **AnonymousAuthentication** and **CertificateAuthentication**:

```
<%IF !AnonymousAuthentication && !CertificateAuthentication%>
  <% FOREACH prompt = prompts %>
    <%NEXT IF !prompt.required %>
    <% prompt.promptText %>
    <input type="<% prompt.type %>" name="<% prompt.name %>" size="20">
  <% END %>
```

Related Topics

- “AnonymousAuthentication” on page 41
- “RealmList” on page 88

changePasswordTitle

Argument to display password change title text.

Object Type Template variable

Included In PasswordChange.thtml,

Usage Optional

Example The following code tests the boolean **showChangePasswordTitle** and, if true, passes the argument to the page, as is.

```
<%IF showChangePasswordTitle %>
  <% changePasswordTitle FILTER verbatim %>
<%END%>
```

Related Topics “showChangePasswordTitle” on page 100

color

	Contains the color definition.
Object Type	Template variable
Default value	None
Included In	Cancel-ppc.thtmlppc.thtmlNewPinSelect.thtmlCancel.thtml, Defender.thtml, ExceededConcurrent.thtml, GeneratePin.thtml, GraceLoginUsed.thtml, LoginPage.thtml, Logout.thtml, NewPin.thtml, NextToken.thtml, PasswordChange.thtml, PasswordExpiration.thtml, PleaseWait.thtml, Remediate.thtml, SM-NewPinSelect.thtml, SM-NewPinSystem.thtml, SM-NewUserPin.thtml, SM- NextToken.thtml, SSL.thtml, SelectRole.thtml, ShowSystemPin.thtml
Usage	Optional
Example	This example sets the background color of the table cell to the value of color . <pre><td bgcolor="<% color %"></td></pre>
Related Topics	None

Continue

	Submits the hidden form when an end-user clicks the Continue button.
Object Type	JavaScript function
Included In	Remediate.thtml
Usage	Optional
Example	The following code fragment defines the Continue button and passes control to the Continue() JavaScript function. <pre><input name="btnContinue" type="button" value="<% textContinue %"> onClick="Continue()"></pre>
Related Topics	"Remediate.thtml" on page 89

cval

	Handles the Radius secure ID challenge value.
Object Type	HTML form field
Included In	Defender.thtml

Usage	Required
Example	<code><INPUT type=hidden name="cval" value="<% secid_challenge %>"></code>
Related Topics	"Defender.shtml" on page 47

Defender.shtml

	Prompts the Radius user to enter his PIN and provides the appropriate challenge values from the server.
Object Type	Template
Usage	Password Management
Required Variables	"secid_challenge" on page 91
Example	None
Related Topics	None

delivery_mode

	Indicates the delivery mode of an applet, whether Java or ActiveX.
Object Type	Template variable
Included In	Logout.shtml
Usage	Required
Example	The following fragment checks to see if the delivery mode is Java when Host Checker or Win32 applets are running during a user logout. If Java is the delivery mode, the code indicates that the page should call the stopComponents function when loading. <pre><% IF (pleasewaitObjectCC pleasewaitObjectHC pleasewaitWin32) && delivery_mode == 'java'%> onload="javascript:stopComponents()" <%END%></pre>
Related Topics	"stopComponents()" on page 109

DoUnload()

	Forces the Infranet Controller to cancel the current mode.
Object Type	JavaScript function

Included In GeneratePin-ppc.shtmlGeneratePin.shtml, NewPin.shtml

Usage Required. You must include this function in the header of **NewPin.shtml** and call it from the body.

Example The following code shows how the **DoUnload()** function appears in the **NewPin.shtml** template. The code serves a similar purpose in other templates, as well.

```
function DoUnload() {  
    if (gCancelNewPinMode) {  
        window.open("secid_cancelpin.cgi", "newpincancel",  
            "resizable=1,height=250,width=200,status=0");  
    }  
}
```

Related Topics

- "NewPin.shtml" on page 75
- "NextToken.shtml" on page 76

DSLlaunchURL

When you deploy the Infranet Controller and *Infranet Enforcer*, users may not know that they must first sign into the Infranet Controller for authentication and endpoint security checking before they are allowed to access a protected resource behind the *Infranet Enforcer*. If captive portal is configured, this variable contains the URL to the protected resource.

Object Type Template variable

Included In AgentInstall.shtml

Usage Required.

Example None

Related Topics

- "AgentInstall.shtml" on page 40

ExceededConcurrent.shtml

Displays a performance warning to the user when too many concurrent users are signed into the Infranet Controller.

Object Type Template

Usage Required. You must always include this template in your zip file. This is a standard Infranet Controller page that displays a performance warning to the user when too many concurrent users are signed into the Infranet Controller. This template does not contain any JavaScript or forms. It does contain an optional link to **starter.cgi**, however, that allows

users to sign into the Infranet Controller as well as error message variables. For more information, see comments in the template.

Required Variables None

Example None

Related Topics None

FinishLoad()

Use this function if you want to allow users to select from multiple realms in the sign-in page.

Object Type JavaScript function

Included In .GeneratePin.thtml, LoginPage.thtml, NewPin.thtml, NextToken.thtml, ShowSystemPin.thtml, Defender.thtml, SM-NewPinSelect.thtml, SM-NewPinSystem.thtml, SM-NewUserPin.thtml, SM-NextToken.thtml.

Usage Optional. If you are authenticating users through a eTrust SiteMinder server using client-side certificate authentication, you may want to hide the standard Infranet Controller sign-in page from users. If you are authenticating users through an anonymous authentication server or certificate authentication server and want to bypass the standard Infranet Controller sign-in page, see “AnonymousAuthentication” on page 41 or “CertificateAuthentication” on page 45.

Example To post data to the Infranet Controller without prompting users for any credentials, you may use the following modified version of the **FinishLoad()** function. Note that this function only bypasses the sign-in page if no errors occur while loading the page:

```
function FinishLoad() {
    recallLastRealmUsed();
    <% IF !LoginPageErrorCode %>
        Login();
        document.frmLogin.submit();
    <% END %>
}
```

Related Topics

- “GeneratePin.thtml” on page 54
- “LoginPage.thtml” on page 65
- “SM-NewPinSelect.thtml” on page 105
- “SM-NewPinSystem.thtml” on page 105
- “SM-NewUserPin.thtml” on page 105
- “SM-NextToken.thtml” on page 106

focus

	Sets the focus on given component on a form.
Object Type	Template variable
Included In	Cancel.shtml, LoginPage.shtml, NewPin.shtml, NextToken.shtml, Defender.shtml, SM-PinSelect.shtml, SM-NewPinSystem.shtml, SM-NewUserPin.shtml, SM-NextToken.shtml
Usage	Required
Example	The following code sets the focus (by way of a JavaScript function also called focus()) to the form object indicated by the template variable focus : <pre>function onLoad() { grab(); document.frmLogin.<% focus %>.focus(); }</pre>
Related Topics	None

FriendlyTime

	Argument in number of seconds.
Object Type	Template variable
Included In	PasswordChange.shtml, PasswordExpiration.shtml
Usage	Argument for error code 2004, Password Too New.
Example	None
Related Topics	• “PageErrorArgs” on page 77 • “PageErrorCode” on page 77

frmCasque

	HTML form that enables an end-user to initiate the CASQUE RADIUS server.
Object Type	HTML form
Included In	Defender.shtml
Usage	Required

Example The following code illustrates the use of the **frmCasque** form. If the plug-in type is the CASQUE RADIUS server plug-in, then the page loads the form:

```
<% IF plugintype == 'CASQUE' %>
  <form name="frmCasque" method="POST" autocomplete=off>
    <INPUT type=hidden name="cval" value="<% secid_challenge %>">
    <INPUT type=hidden name="p" value="casqueapp">
    <a href="#" onClick="submitFrmCasque(); return false;">
      LAUNCH CASQUE PLAYER
  </form>
```

Related Topics “submitFrmCasque()” on page 110

frmLogin

HTML form that enables the template page to trigger a call to the **Login()** function and begins the sign in process for a user.

Object Type HTML Form

Included In LoginPage.html

Usage Required

Example The following code is the form definition statement:

```
<form name="frmLogin" action="login.cgi" method="post" autocomplete="off"
onsubmit="return Login()">
```

Related Topics “autocomplete” on page 43

frmNextToken

Next Token HTML form.

Object Type HTML Form

Included In NextToken.shtml, SM-NextToken.shtml

Usage Required

Example The following code illustrates the suggested form definition:

```
<FORM name="frmNextToken" action="login.cgi" method="POST"
autocomplete=off onsubmit="return SubmitClicked();">
```

Related Topics “NextToken.shtml” on page 76

frmSelectRoles

	Select Roles HTML form
Object Type	HTML Form
Included In	SelectRole.thtml
Usage	Required
Example	The following code illustrates the suggested form definition: <FORM name="frmSelectRoles" autocomplete=off>
Related Topics	"SelectRole.thtml" on page 95

g_time

	The time when the browser first loads the page.
Object Type	JavaScript variable
Included In	PleaseWait.thtml
Usage	Optional
Example	None
Related Topics	"PleaseWait.thtml" on page 83

gCancelNewPinMode

	Prevents the Infranet Controller from cancelling Generate PIN mode while the user is trying to save a PIN or generate a new one.
Object Type	JavaScript variable/flag
Included In	GeneratePin.thtml, NewPin.thtml
Usage	Required. You must include this flag in the header of GeneratePin.thtml . Works together with the JavaScript DoUnload() and SubmitClicked() functions to call the appropriate CGI on the Infranet Controller if the user chooses to close the browser window without cancelling his session first. If you do not include this code, the user may be locked out of the Infranet Controller when he closes his browser. In addition to including this JavaScript in your header, you must also call the functionDoUnload() function from the body of GeneratePin.thtml.

Example You include the following code in the header of the **NewPin.html** template. This function saves or cancels a PIN per the user's request. Note that in this example, the **gCancelNewPinMode** is set to false though it was initialized to true when the page loaded.

```
// Save / Cancel a PIN
function SubmitClicked() {
  gCancelNewPinMode = false;
  return true;
}
```

Related Topics "SubmitClicked()" on page 109

gCancelNextTokenMode

Prevents the Infranet Controller from cancelling Next Token mode while the user is trying to enter his token.

Object Type JavaScript function

Included In NextToken.html

Usage Required. The header in **NextToken.html** contains two required JavaScript functions, **DoUnload()** and **SubmitClicked()**, as well as the **gCancelNextTokenMode** flag. These work together to call the appropriate CGI on the Infranet Controller if the user chooses to close the browser window without cancelling his session first. If you do not include this code, the user may be locked out of the Infranet Controller when he closes his browser. In addition to including this JavaScript in your header, you must also call the **DoUnload()** function from the body of **NextToken.html**.

Example You must include this flag in the header of **NextToken.html**:

```
var gCancelNextTokenMode = true;
```

Related Topics "NextToken.html" on page 76

geckoBrowser

Boolean set to 1 if the browser is gecko-based (Firefox).

Object Type Template variable

Included In Logout.html, PleaseWait.html

Usage Required

Example The following code checks to see if the browser is Firefox. If yes, then the page loads the proxy form in an **iFrame** (inline frame).

```
<% IF geckoBrowser %>
  <IFRAME ID="proxy_frm" src="/dana-na/help/blank.html" WIDTH=0
    HEIGHT=0 FRAMEBORDER=0 SCROLLING=NO></IFRAME>
<% END %>
```

Related Topics "safari" on page 90

GeneratePin.shtml

Enables the user to create a system-generated PIN.

Object Type Template

Usage Required

Required Variables None

Example When configuring the `secid_pinselectmode` variable (which sets the user's PIN select mode), note that possible values include:

Table 6: PIN usage indicators

Mode	Description
0	User must use the system-generated PIN. He cannot enter his own PIN.
1	User may enter his own PIN or use the system-generated PIN.
2	User must enter his own PIN. He cannot use the system-generated PIN.

Related Topics None

GetCookieValue()

Generic helper function that retrieves a value from a cookie, creates a string the length of the value, and populates the string with the value.

Object Type JavaScript function

Included In `LoginPage.shtml`, `PleaseWait.shtml`

Usage Use this function if you want to allow users to select from multiple realms in the sign-in page.

sName—Cookie name.

Example The function definition follows:

```
function GetCookieValue(sName) {
    var s = document.cookie;
    sName += "=";

    // where nv pair starts
    var nStart = s.indexOf(sName);
    if (nStart == -1)
        return "";
    else
        nStart += sName.length;
    // if more values, clip, otherwise just get rest of string
    var nEnd = document.cookie.indexOf(";", nStart);
    if (nEnd == -1)
        s = unescape(s.substring(nStart));
    else
        s = unescape(s.substring(nStart, nEnd));
    return s;
}
```

- Related Topics**
- “LoginPage.thtml” on page 65
 - “PleaseWait.thtml” on page 83

GraceLoginsRemaining

Number of remaining grace logins.

Object Type Template variable

Included In GraceLoginUsed.thtml

Usage Optional. Argument for error code 2016.

Example The following code shows an example of how to display a message for the number of grace logins available to the user:

```
<% IF LoginPageErrorCode == 2016 %>
    You have <% LoginPageErrorArgs.GraceLoginsRemaining %> grace logins.
<% END %>
```

- Related Topics**
- “GraceLoginUsed.thtml” on page 55
 - “PageErrorArgs” on page 77
 - “PageErrorCode” on page 77

GraceLoginUsed.thtml

Tells the user how many more times he can sign in using his current username and password.

Object Type Template

Usage	Required
Required Variables	"startPageLink" on page 108
Example	None
Related Topics	None

HC_REMED_POLICIES_CHECK

	Default text to suggest that policies are being evaluated.
Object Type	Template variable
Included In	Remediate.shtml
Usage	Optional
Example	The following code checks to see if the page is loading. If yes, then the page displays several messages, including the default text to suggest that the Infranet Controller is evaluating policies: <pre><% IF showLoading %> <% loading %> <% pleasewait %> <% HC_REMED_POLICIES_CHECK %> <% END %></pre>
Related Topics	• "HC_REMED_SHOW_ADV_PREFS" on page 56 • "Remediate.shtml" on page 89

HC_REMED_SHOW_ADV_PREFS

	Default text instructing how to re-enable remediation.
Object Type	Template variable
Included In	Remediate.shtml
Usage	Optional
Example	The following code illustrates how to call the variable to display instructions to re-enable remediation: <pre><% IF showRemedOption %> <input name="btnHideRemed" type="button" value="<% textRemedOption %>" onClick="HideRemed();</pre>

```

    <% disabled %>">
    <% HC_REMED_SHOW_ADV_PREFS %>
<% END %>

```

- Related Topics**
- “HC_REMED_SHOW_ADV_PREFS” on page 56
 - “Remediate.html” on page 89

hcInAcTimeout

Specifies the Host Checker login inactivity in minutes.

Object Type Template variable

Included In LoginPage.html

Usage Required

Example The following code sets the timeout interval to 60000 milliseconds times the smaller value of cclnAcTimeout or hcInAcTimeout.

```

<% IF hcInAcTimeout > cclnAcTimeout %>
    window.setTimeout("deletepreauth()", 60000 * <% cclnAcTimeout %>);
<% ELSE %>
    window.setTimeout("deletepreauth()", 60000 * <% hcInAcTimeout %>);
<% END %>

```

Related Topics “LoginPage.html” on page 65

hcRunning

Specifies whether or not Host Checker is currently running.

Object Type Template variable

Included In LoginPage.html

Usage Required

Example The following code sets the timeout interval based on whether Host Checker and Cache Cleaner are both running or only Host Checker.

```

<% IF hcRunning && ccRunning %>
    <% IF hcInAcTimeout > cclnAcTimeout %>
        window.setTimeout("deletepreauth()", 60000 * <% cclnAcTimeout %>);
    <% ELSE %>
        window.setTimeout("deletepreauth()", 60000 * <% hcInAcTimeout %>);
    <% END %>
<% ELSIF hcRunning %>

```

```
window.setTimeout("deletepreauth();", 60000 * <% hcinAcTimeout %>);  
<% END %>
```

Related Topics "LoginPage.thtml" on page 65

heading

Contains default text for the different remediation headings.

Object Type Template variable

Included In Remediate.thtml, Remediate-ppc.thtml

Usage Required

Example The following code shows a heading and a message if the **showHeading** variable is true:

```
<% IF showHeading %>  
<% heading %>  
<% msg FILTER verbatim %>  
<% END %>
```

Related Topics "showHeading" on page 102

help

String to use as the button name.

Object Type Template variable

Default value "Help"

Included In LoginPage.thtml

Usage Optional

Example The string "Help" (the default value of the **help** variable) is displayed on the button.

```
<input type='submit' name='help' value="<% help %>"
```

Related Topics "Adding Custom Help Files" on page 10

help_on

Flag that determines whether the help button should be displayed.

Object Type Template variable

Default value 0 (integer)

Included In	Loginpage.thtml
Usage	Optional
Example	The following code displays a help button if the help_on variable is true. <pre><% IF help_on %> <input type='submit' name='help' value="<% help %>" onclick='window.open("welcome.cgi?p=help", "wndHelp", "height=400,width=500,resizeable=yes,scrollbars=yes"); return false;'></pre>
Related Topics	"Adding Custom Help Files" on page 10

HideRemed()

	Submits the hidden form when the end-user clicks the Hide Remediation button.
Object Type	JavaScript function
Included In	Remediate.thtml, Remediate-ppc.thtml
Usage	Required
Example	The following code calls the HideRemed() function: <pre><% IF showRemedOption %> <input name="btnHideRemed" type="button" value="<% textRemedOption %>" onClick="HideRemed(); <% disabled %>"> <% HC_REMED_SHOW_ADV_PREFS %> <% END %></pre>
Related Topics	"Remediate.thtml" on page 89

Home

	References the top-level directory in the zip file.
Object Type	Template variable
Included In	All templates
Usage	You can use this variable or the double-dot (..) convention to reference the top-level directory.
Example	Both of the following references are valid: <pre><% Home %>/images/logo.gif ../images/logo.gif</pre>

Related Topics “Using the Template Toolkit” on page 10

ie_winxp

Boolean indicating if the browser is Internet Explorer and the OS is Windows XP.

Object Type Template variable

Included In PleaseWait.thtml

Usage Optional. The variable value is 1 if the browser is Internet Explorer and the OS is Windows XP.

Example <% IF ie_winxp %>

Related Topics “PleaseWait.thtml” on page 83

instructions

Instructional text displayed to the user on the login page.

Object Type Template variable

Default Value “Please sign in to begin your secure session”

Included In LoginPage.thtml

Usage Optional

Example <% instructions FILTER verbatim %>

Related Topics None

isInfranetAgent

Boolean indicating whether or not the request is from an Odyssey Access Client 1.x agent.

Object Type Template variable

Included In PleaseWait.thtml

Usage Optional

Example <% isInfranetAgent %>

Related Topics “PleaseWait.thtml” on page 83

isLinux

	Boolean indicating whether or not the end-user's operating system is a Linux system.
Object Type	Template variable
Included In	PleaseWait.thtml
Usage	Optional
Example	<% isLinux %>
Related Topics	"PleaseWait.thtml" on page 83

isSAMEnabled

	Boolean indicating whether or not JSAM application is enabled and running.
Object Type	Template variable
Included In	Logout.thtml
Usage	Optional. Allows the end-user to close the JSAM window.
Example	The following code checks to see if JSAM is enabled and running. If it is, then the system uses the JavaScript below to close the JSAM window: <pre> <% IF isSAMEnabled %> <!-- terminate JSAM --> <SCRIPT LANGUAGE="JavaScript"> var win = "<%user%>"; win = win.replace(/[^0-9a-zA-Z]/g,"a"); w = window.open("", win ,"height=1,width=1"); if (w) { w.bSessionClose = true; w.close(); } </SCRIPT> <% END %> </pre>
Related Topics	"Logout.thtml" on page 69

keyboard.js

	Enables keyboard types of functions for a mouseless environment.
Object Type	JavaScript file

Included In	LoginPage.shtml (Kiosk)
Usage	Required in Kiosk zip. This template displays an error if the user signs out, if the user's session times out, or if the Infranet Controller uninstalls Host Checker from the user's system. For more detailed information, see "Logout.shtml" on page 69. You must always include this template in your Kiosk.zip file.
Example	None
Related Topics	"LoginPage.shtml" on page 65

labelInstructions

	Instructions tag, indicating a location for instructions to appear on the page.
Object Type	Template variable
Included In	Remediate.shtml, Remediate-ppc.shtml
Usage	Optional
Example	The following code loops through a list of failed policies, displays each policy name and remediation instructions: <pre><% FOREACH failedPolicy = listFailedPolicies %> <% loop.count %> <% failedPolicy.name %> <% labelInstructions %> <% failedPolicy.remediation FILTER verbatim%> <% END %></pre>
Related Topics	"Remediate.shtml" on page 89

listFailedPolicies

	Template Toolkit hash value of failed policies (keys: name and remediation).
Object Type	Template variable
Included In	Remediate.shtml
Usage	Required
Example	The following example checks to see if the page should show failed policies and, if so, loops through the list of failed policies (the hash list listFailedPolicies) and displays the name of each policy along with label instructions and remediation instructions: <pre><% IF showPolicies %></pre>

```

<% FOREACH failedPolicy = listFailedPolicies %>
<% loop.count %>
<% failedPolicy.name %>
<% labelInstructions %>:
<% failedPolicy.remediation FILTER verbatim%>
<% END %>

```

Related Topics "Remediate.thtml" on page 89

loading

Contains default text to be printed when ActiveX/applet is loading.

Object Type Template variable

Included In Remediate.thtml

Usage Optional

Example The following code tests to see if the page is still loading and displays messages to the end-user:

```

<% IF showLoading %>
<% loading %>
<% pleasewait %>

```

Related Topics "Remediate.thtml" on page 89

locale

Returns a locale that applies to an applet the page is attempting to upload.

Object Type Template variable

Included In Logout.thtml, MeetingAppletInstaller.thtml, MeetingTrouble.thtmlLogout.thtml

Usage Required

Example The following code sets the HTML form locale based on the template variable:

```

<PARAM NAME="locale" VALUE="<% locale %>">

```

Related Topics "safari" on page 90

loginmode

Passes the user's mode to the server using a hidden value.

Object Type HTML form field

Included In	GeneratePin.shtml, NewPin.shtml, NextToken.shtml, Defender.shtml, SM-NewPinSelect.shtml, SM-NewPinSystem.shtml, SM-NewUserPin.shtml, SM-NextToken.shtml
Usage	Required. Typically names a template directive that returns the user's login mode from a server.
Example	The following code assigns the secure ID loginmode to the form field: <INPUT type=hidden name="loginmode" value="<% secid_loginmode %>">
Related Topics	"secid_loginmode" on page 92

Login()

	Verifies that realms exist, remembers currently selected authentication realms, and establishes the user's time zone.
Object Type	JavaScript function
Included In	LoginPage.shtml
Usage	Required. This function includes two actions: • Remember the selected authentication realm—Use this action within the function if you want to allow users to select from multiple realms in the sign-in page. • Establish the user's time zone—Required. The function uses the tz_offset (time zone offset) variable and getTimezoneOffset() function to determine the user's time zone.
Example	<pre>function Login() { // Remember currently selected auth realm if (document.frmLogin.realm != null && document.frmLogin.realm.type == "select-one") { SetLastRealm(document.frmLogin.realm.options [document.frmLogin.realm.selectedIndex].text); } if (document.frmLogin.tz_offset != null) { var wdate = new Date (95, 12, 1); var sdate = new Date (95, 6, 1); var winter = (-1) * wdate.getTimezoneOffset(); var summer = (-1) * sdate.getTimezoneOffset(); document.frmLogin.tz_offset.value = winter < summer ? winter : summer; } return true; }</pre>
Related Topics	"LoginPage.shtml" on page 65

LoginPage.thtml

Calls the RSA Soft ID client, enabling the user to sign into the Infranet Controller using Soft ID authentication.

Object Type Template

Usage Required. You must always include this template in your zip file. You must always include **LoginPage.thtml** in your zip file, even if you are using an authentication server that does not require a username or password. This is the standard Infranet Controller sign-in page that collects the user's name, password, and authentication realm, and displays an error if authentication fails. For information about hiding this page from users authenticated against a(n):

- Anonymous server, see "AnonymousAuthentication" on page 41
- Certificate server, see "CertificateAuthentication" on page 45
- eTrust SiteMinder server using client-side certificate authentication, see "Login()" on page 64

LoginPage.thtml contains the following suggested form definition. In this form definition, most elements are required:

```
<form name="frmLogin" action="login.cgi" method="post" autocomplete="off"
onsubmit="return Login()">
```

Required Variables "LoginPageErrorMessage" on page 69

Example None

Related Topics • "Login()" on page 64

LoginPageErrorArgs

Contains the arguments for an error or message.

Object Type Template variable

Included In GraceLoginUsed.thtml, PasswordExpiration.thtml

Usage Optional

Example In the following example, **GraceLoginsRemaining** is the argument to error code 2016. The directive that contains **LoginPageErrorArgs.GraceLoginsRemaining** returns the number of remaining login attempts the Infranet Controller allows a user.

```
<% IF LoginPageErrorCode == 2016 %>
  Your password has expired. A grace login was used.
```

```

    You have <% LoginPageErrorArgs.GraceLoginsRemaining %> grace logins
    remaining.
<% ELSE %>
<% LoginPageErrorMessage %>
<% END %>

```

Related Topics “LoginPageErrorCode” on page 66

LoginPageErrorCode

Codes indicating errors that you can display to users.

Object Type Template variable

Included In ExceededConcurrent.html, GraceLoginUsed.html, LoginPage.html, PasswordExpiration.html, SSL.html

Usage Optional. Although this template variable is optional, you should leave intact the template variables that are already included in the various templates. You can modify many of the error messages, but you must leave the codes as defined. The Infranet Controller returns these codes on various conditions. Some may be no more than confirmation messages. Others are conditions to which the user needs to respond.

Example The following example shows how you can use the Template Toolkit conditional statements to define two alternate error conditions. In this example, the different error codes display appropriate HTML pages. If the Infranet Controller does not return either of the error codes, the user never sees the two error pages.

```

<head>
  <% IF LoginPageErrorCode == 1021%>
    <meta http-equiv=" refresh" content=" 0;url=AcctDisabledError.html" >

```

Table 7 on page 66 lists possible errors that the user may see on this page as well as the corresponding text returned by the **LoginPageErrorMessage** variable:

Table 7: Login Error Messages, Codes, and Related Notes

Error message	Error Code	Comments
Invalid Username or Password	1002	Wrong user credentials.
This login requires a digital certificate.	1003	Client certificate not passed for authentication.
The digital certificate is invalid.	1004	
Only admins are permitted to login.	1006	
Logins are not permitted from this IP address.	1007	The source IP failed the IP Restriction check.

Table 7: Login Error Messages, Codes, and Related Notes (*continued*)

Error message	Error Code	Comments
Logins are not permitted using this browser	1008	The User Agent header failed the User Agent Restriction check.
There are no auth servers defined for this user.	1009	Displayed if a realm is not sent to the Infranet Controller. This error is only displayed if a coding error exists in LoginPage.html.
Exceeded the number of concurrent users.	1010	Exceeded the number of concurrent users based on the user license plus a 10% allowance.
The IP has been blocked due too many concurrent sign-in attempts.	1011	
The password is too short.	1012	The password failed the Password Restriction check. When using eTrust auto-signon, the end-user should never see this event.
Not using SSLv3	1015	This site requires Secure Sockets Layer (SSL) protocol version 3.
This site requires Strong ciphers.	1016	
Too many users have logged in	1017	The number of concurrent users signed in to the system has exceeded the system limit but is still within the 10% allowance.
Sign on for administrators is disabled. You can try again in a few minutes.	1018	Displayed when a super administrator is signed in and the Infranet Controller is in recovery mode.
Your New PIN has been saved. Please make sure to remember it.	1019	Only used with ACE authentication.
Password Change Success	1020	Used with the password management feature.
Account Disabled	1021	Used with the password management feature.
Account Locked Out	1022	Used with the password management feature.

Table 7: Login Error Messages, Codes, and Related Notes (*continued*)

Error message	Error Code	Comments
Account Expired	1023	Used with the password management feature.
Infranet Controller session has been terminated.	1024	End-user explicitly logged off.
You do not have permission to login.	1025	This event can be triggered when the client machine does not pass any realm restrictions (hence, no realm), or does not satisfy any end-point security policies (HC/CC).
This realm has reached the max session limit.	1027	Exceeded maximum number of users for this authentication realm.
Unlicensed Feature	1028	License required to use this access method is missing.
Access denied. Please try signing in again using a host name (for example, https://department.yourcompany) instead of an IP address (such as http://10.11.12.13)	1029	Only used with eTrust SiteMinder authentication.
You do not have the correct privileges to access the system. Please contact your administrator for more information.	1030	The end-user is not allowed to sign in because the role mapping rules of the authentication realm do not map the end-user to any role.
Certificate has been revoked.	1032	The client certificate has been revoked.
Cannot determine the status of your certificate.	1033	The Infranet Controller encountered a failure while performing an OCSP check.
You must use your source site's Intersite Transfer Service to access this resource.	1034	The Infranet Controller cannot access any SAML assertion while performing SAML SSO. The user is prompted to access the source site's Intersite Transfer Service to continue.
The number of concurrent Advanced Endpoint Defense (Malware Protection) users signed into the system has exceeded the system limit.	1035	This event will be triggered only when the feature is turned ON. The system limit is built-in 25 user license plus any purchased Advanced Endpoint Defense user license.

Related Topics “LoginPageErrorMessage” on page 69

LoginPageErrorMessage

Infranet Controller error message text that you can display to users.

Object Type Template variable

Included In ExceededConcurrent.thtml, GraceLoginUsed.thtml, LoginPage.thtml, Logout.thtml, PleaseWait.thtml, PasswordExpiration.thtml, SSL.thtmlUsage

Required. This text corresponds with a code returned by **LoginPageErrorCode**. Note that you cannot change this text on the Infranet Controller, but you can include code in your template to display different text based on the error code returned by the **LoginPageErrorCode** variable. You can also use the variable in a directive to display an error message in a particular location or based on a given event.

Example The following code prompts the user to re-enter credentials or to sign in at another time. Note that the messages can be whatever you want them to be:

```
<% IF LoginPageErrorCode == 1002 %>
  <b> Your username or password is incorrect. Please reenter your credentials.
</b>
<%ELIF LoginPageErrorCode == 1006 %>
  <b> The system is undergoing maintenance. Only administrators are allowed
to sign in at this time.</b>
<% END %>
```

The following code tests the pleasewait condition and, if true, displays the Login page error message that occurs as a result:

```
if (document.getElementById('pleasewait1')) {
  document.getElementById('pleasewait1').innerHTML =
    "<% LoginPageErrorMessage FILTER verbatim%>";
}
```

Related Topics “LoginPageErrorCode” on page 66

Logout.thtml

Displays an error if the user signs out, if the user’s session times out, or if the uninstalls Host Checker or Cache Cleaner from the user’s system. This is a standard Infranet Controller page that displays an error if the user signs out of an agentless deployment, if the user’s session times out, or if the Infranet Controller uninstalls Host Checker from the user’s system.

Object Type Template

Usage Required. You must always include **Logout.thtml** in your zip file.

This template contains JavaScript for detecting, stopping, installing, and uninstalling the Host Checker and Cache Cleaner components, images and text to display to users while the performs these actions, and JavaScript that closes and opens the J-SAM window. This template contains JavaScript for detecting, stopping, installing, and uninstalling Host Checker components, as well as images and text to display to users while the Infranet Controller performs these actions.

This template also includes variables telling users to wait while components install, a link that users can use to sign back into the Infranet Controller, and error message variables.

When configuring the **LoginPageErrorMessage** variable included in this template, note that the text that the Infranet Controller displays to users corresponds with a code returned by **LoginPageErrorCode**. You cannot change this text on the Infranet Controller, but you can include code in your template to display different text based on the error code returned by the **LoginPageErrorCode** variable. For example:

```
<% IF LoginPageErrorCode == 1001 %>
  <b> Your secure gateway session has ended due to inactivity.</b>
<% END %>
```

Described below are the possible errors that the **LoginPageErrorCode** may return as well as the corresponding text returned by the **LoginPageErrorMessage** variable:

Table 8: Additional Login Error Messages and Codes

Error Message	Error Code
Maximum Session Timeout Reached.	1000
Idle Time Out.	1001

For detailed information about the JavaScript and variables described here, see comments in the template.

- Required Variables**
- “delivery_mode” on page 47
 - “geckoBrowser” on page 53
 - “locale” on page 63
 - “LoginPageErrorMessage” on page 69
 - “Netscape4xx” on page 74
 - “pleasewaitLogoutJSCode” on page 84
 - “PleaseWaitObjectEC” on page 84
 - “pleasewaitWin32” on page 85
 - “safari” on page 90
 - “Setupappversion” on page 99

Example None

Related Topics None

MinimumPasswordLength

Argument for error code 2006. The argument is the minimum length of password required.

Object Type Template variable

Included In PasswordChange.shtml, PasswordChange-ppc.shtml

Usage Optional

Example None

Related Topics

- “LoginPageErrorCode” on page 66
- “PasswordChange.shtml” on page 80

msg

Contains any extra remediation related default instructions.

Object Type Template variable

Included In Remediate.shtml

Usage Optional

Example The following code checks to see if a heading is to be shown. If yes, then the page displays the heading, followed by any remediation message that might be available.

```
<% IF showHeading %>
<% heading %>
<% msg FILTER verbatim %>
<% END %>
```

Related Topics “Remediate.shtml” on page 89

netacekey

Passes the user’s eTrust ACE key to the server using a hidden value.

Object Type HTML form field

Included In SM-NewUserPin.shtml, SM-NewPinSelect.shtml, SM-NextToken.shtml

Usage Required. The form assigns the **netesecid_key** value to **netacekey**.

Example The following example shows the use of **netacekey** in the **INPUT** statement:

```
<INPUT type=hidden name="netacekey" value="<% netesecid_key %>">
```

Related Topics “netesecid_key” on page 72

netesecid_key

Contains the user's eTrust ACE key.

Object Type Template variable

Included In **SM-NewPinSelect.thtml**, **SM-NextToken.thtml**, **SM-NewPinSystem.thtml**, **SM-NewUserPin.thtml**

Usage Required. The value is passed by the page to the server.

Example The following example shows the use of **netesecid_key** in the **INPUT** statement:

```
<INPUT type=hidden name="netacekey" value="<% netesecid_key %>">
```

Related Topics “netacekey” on page 71

netesecid_loginmode

Returns a value of Next Token mode or New PIN mode.

Object Type Template variable

Included In **SM-NewPinSelect.thtml**, **SM-NextToken.thtml**, **SM-NewPinSystem.thtml**, **SM-NewUserPin.thtml**

Usage Required. You must send this value to the server, but if you do not want to expose it to the user, you may send it as a hidden form field.

Example The following hidden **INPUT** statement sets the login mode based on the **netesecid_loginmode** variable.

```
<INPUT type=hidden name="loginmode" value="<% netesecid_loginmode %>">
```

Related Topics “loginmode” on page 63

netesecid_pinerr

Error code and corresponding message that informs the user that he mis-entered his PIN.

Object Type	Template variable
Included In	SM-NewUserPin.thtml
Usage	Optional
Example	The following code checks the value of <code>netesecid_pinerr</code>, a boolean, to see if the PIN the user enters matches the PIN on the authentication server: <pre> <% IF netesecid_pinerr == 0 %> New PIN Assignment <% ELSIF netesecid_pinerr == 1 %> The Two PINs Entered Do Not Match <% END %> </pre>
Related Topics	"SM-NewUserPin.thtml" on page 105

netesecid_realm

	The user's realm.
Object Type	Template variable
Included In	SM-NewPinSelect.thtml, SM-NextToken.thtml, SM-NewPinSystem.thtml, SM-NewUserPin.thtml
Usage	Optional
Example	The following code shows the configuration of the page based on various server-provided values: <pre> <FORM name="frmNewPin" action="login.cgi" method="POST" autocomplete=off> <INPUT type=hidden name="loginmode" value="<% netesecid_loginmode %>"> <INPUT type=hidden name="realm" value="<% netesecid_realm %>"> <INPUT type=hidden name="neteacekey" value="<% netesecid_key %>"> </pre>
Related Topics	"SM-NewUserPin.thtml" on page 105 "SM-NextToken.thtml" on page 106

netesecid_username

	The user's username.
Object Type	Template variable
Included In	SM-NewPinSelect.thtml, SM-NextToken.thtml, SM-NewPinSystem.thtml, SM-NewUserPin.thtml

Usage	Optional
Example	The following INPUT statement assigns the netesecid_username to the form username field: <pre><INPUT type="text" name="username" value="<% netesecid_username %>"></pre>
Related Topics	"netesecid_realm" on page 73

netesecidactionCancel

	Cancels a eTrust secure ID action currently in progress.
Object Type	HTML form field
Included In	SM-NewPinSelect.thtml, SM-NextToken.thtml, SM-NewPinSystem.thtml, SM-NewUserPin.thtml
Usage	Required. The value of either this field or of netesecidactionEnter must be sent to the server: <pre><INPUT type="submit" value="Cancel" name="netesecidactionCancel"></pre>
Related Topics	"netesecidactionEnter" on page 74

netesecidactionEnter

	Initiates the submission of the current eTrust secure ID action.
Object Type	HTML form field
Included In	SM-NewPinSelect.thtml, SM-NextToken.thtml, SM-NewPinSystem.thtml, SM-NewUserPin.thtml
Usage	Required. The value of either this field or of netesecidactionCancel must be sent to the server.
Example	<pre><INPUT type="submit" value="Enter" name="netesecidactionEnter"></pre>
Related Topics	"netesecidactionCancel" on page 74

Netscape4xx

	Checks for Netscape version 4.x browser.
Object Type	Template variable

Included In Logout.thtml

Usage Required. Checks to see if the user's browser is Netscape version 4.x. If so, then, on logout, error messages are passed to the page in a compatible way.

Example None

Related Topics "Logout.thtml" on page 69

NewPin.thtml

Prompts ACE users to enter a new PIN or create a system-generated PIN before signing into the Infranet Controller.

Object Type Template

Usage Required. When configuring the **secid_pinselectmode** variable (which sets the user's PIN select mode), note that possible values include:

Table 9: PIN Error Messages

Mode	Description
0	User must use the system-generated PIN. He cannot enter his own PIN.
1	User may enter his own PIN or use the system-generated PIN.
2	User must enter his own PIN. He cannot use the system-generated PIN.

When configuring the **secid_pinserr** variable (which stores the error code and message that informs the user if he mis-entered his PIN), note that possible values include:

Table 10: Additional PIN Error Messages

Code	Value
0	New PIN is required.
1	The two PINs entered do not match.
2	PIN Format is invalid.
3	PIN length is invalid.

Required Variables

- "secid_contextid" on page 91
- "secid_loginmode" on page 92

Example None

Related Topics None

NextToken.shtml

Prompts the user to verify his credentials by entering his SecureID token code.

Object Type Template

Usage Optional. Although this template is optional, you should always include it in the zip file that you upload to the Infranet Controller.

Required Variables

- "secid_contextid" on page 91
- "secid_loginmode" on page 92

Example None

Related Topics None

onsubmit

Event trigger in a form.

Object Type HTML form element

Included In GeneratePin.shtml, LoginPage.shtml, NewPin.shtml, NextToken.shtml, SM-NewUserPin.shtml

Usage Required if you are using Secure Meeting) Returns the time zone offset value set by the function.Returns the time zone offset value set by the **Login()** function.

Example The following code shows a typical use of onsubmit:

```
onsubmit="return Login()"
```

Related Topics None

p

Specifies the page name as CASQUEAPP.

Object Type HTML form field

Included In Defender.shtml

Usage Required

Example The following code assigns the value CASQUEAPP to the form **INPUT** field p:

```
<INPUT type=hidden name="p" value="casqueapp">
```

Related Topics “secid_challenge” on page 91

PageErrorArgs

Contains the arguments for an error or message.

Object Type Template variable

Included In PasswordChange.thtml

Usage Optional

Example The following code checks to see if a password length error occurs, then displays an informational message:

```
<% IF PageErrorCode == 2015 %>
  New password must not be longer than
  <% PageErrorArgs.MaximumPasswordLength %> characters.
```

Related Topics

- “PageErrorCode” on page 77
- “PasswordChange.thtml” on page 80

PageErrorCode

Contains the error code in case of an error or message.

Object Type Template variable

Included In LoginPage.thtml, PasswordChange.thtml

Usage Optional

Table 11: Page Error Codes

Error Code	Error message
1020	Password Change Success
1021	Account Disabled
1022	Account Locked out
1023	Account Expired
2001	Password Mismatch

Table 11: Page Error Codes (*continued*)

Error Code	Error message
2002	Invalid Password
2003	Password Change not allowed
2004	Password too new
2006	Password too short
2007	Password too trivial
2008	Password exists in password history
2009	Password Expired Message
2010	Password must change at next logon message
2011	Password does not meet complexity requirements
2012	Could not connect to auth server to change password
2013	Password change failed
2014	Password successfully changed
2015	Password will expire in some amount of time
2016	You have some number of grace logins.

For more errors and messages, see “LoginPageErrorCode” on page 66 and “LoginPageErrorMessage” on page 69.

Example You cannot change the messages that appear in Table 11 on page 77, but you can create your own messages and include them in your customized pages. For example:

```
<% IF PageErrorCode == 2009 %>
  Your password has expired. Please contact your system administrator.
<% END %>
```

- Related Topics**
- “LoginPageErrorCode” on page 66
 - “LoginPageErrorMessage” on page 69

password

Indicates the password for an optional JavaScript function in the **Defender.thtml** template.

Object Type JavaScript variable

Included In	Defender.shtml
Usage	Optional
Example	The header of Defender.shtml contains the following optional function to automatically place the cursor in the password field of the Defender page. If you choose to include this function in the header, you must also call it from the body of Defender.shtml . Also note that you should not change the JavaScript function name, form name, or form field name and that the form and form fields should have the same name.
	<pre>function FinishLoad() { document.frmDefender.password.focus(); }</pre>
Related Topics	"FinishLoad()" on page 49

password

	Sets an HTML form field as a password field.
Object Type	Input form field
Included In	LoginPage.shtml, NewPin.shtml, NextToken.shtml, Defender.shtml, PasswordChange.shtml, SM-NewPinSelect.shtml, NewPinSystem.shtml, SM- NewUserPin.shtml, SM-NextToken.shtml, GeneratePin.shtml
Usage	This field is required for all types of authentication servers except anonymous servers and certificate servers.
Example	The following code assigns the password field:
	<pre><input type="password" name="password"></pre>
Related Topics	"LoginPage.shtml" on page 65 2

password2

	Verified password.
Object Type	HTML form field
Included In	NewPin.shtml, SM-NewPinSelect.shtml, SM-NewUserPin.shtml, SM-NextToken.shtml, GeneratePin.shtml
Usage	Required. The password2 and password fields pass the password and verified password that the user enters to the server for verification.

Example The value of the following fields must match:

```
<INPUT type="password" name="password">  
<INPUT type="password" name="password2">
```

Related Topics "password" on page 79

PasswordChange.shtml

Notifies the user that his password is due to expire and prompts him to change it.

Object Type Template

Usage Optional. When you include the **changePasswordTitle** variable in your template, the Infranet Controller adds a title to the standard old/new/confirm password text boxes specifying whether the user must change his password.

The **showChangePasswordTitle** variable indicates whether or not to show the change password text. The variable value is true if primary authentication exists in the login process. Without the **changePasswordTitle** variable, the password change page only displays the old/new/confirm password text boxes and a message above, suggesting that the user change password. With the **changePasswordTitle** variable, the password change page displays the old/new/confirm password text boxes, a message above suggesting that the user change password, and another heading at the top of that page, stating **Change Primary (or Secondary) Password**.

Required Variables None

Example None

Related Topics

- "changePasswordTitle" on page 45
- "showChangePasswordTitle" on page 100

Example None

Related-Topics "PasswordChange.shtml" on page 80

PasswordComplexityMessage

The password complexity message.

Object Type Template variable

Included In PasswordChange-ppc.shtml, PasswordChange.shtml

Usage Optional. Argument for error code 2011.

Example The following code notifies the user that the password does not meet complexity requirements:

```
<% IF PageErrorCode == 2011 %>  
  New password not complex enough.  
  Must consist of <% PageErrorArgs.PasswordComplexityMessage %>  
  characters.  
<% END %>
```

- Related Topics**
- “PageErrorArgs” on page 77
 - “PageErrorCode” on page 77

PasswordExpiration.thtml

Notifies the user that his password has expired and prompts him to change it.

Object Type Template

Usage Optional

Required Variables “startPageLink” on page 108

Example None

Related Topics None

PasswordExpirationFriendlyTime

Message displaying how soon password expires.

Object Type Template variable

Included In PasswordExpiration.thtml

Usage Argument for error code **2015**. Shows time in days, hours, and minutes.

Example The following code checks for error code **2015**, and displays a message containing the time remaining before the password expires:

```
<% IF LoginPageErrorCode == 2015 %>  
  Password expires in  
  <% LoginPageErrorArgs.PasswordExpirationFriendlyTime %>.  
<% END %>
```

- Related Topics**
- “LoginPageErrorArgs” on page 65
 - “LoginPageErrorCode” on page 66
 - “PasswordExpirationSeconds” on page 82

PasswordExpirationSeconds

	The number of seconds in which the current user's password will expire.
Object Type	Template variable
Included In	PasswordExpiration.thtml
Usage	Optional. Argument for error code 2015.
Example	The following code indicates that the user's password will expire in a certain number of seconds: <pre><% IF LoginPageErrorCode == 2015 %> Password will expire in <% LoginPageErrorArgs.PasswordExpirationSeconds %> seconds. <% END %></pre>
Related Topics	• "LoginPageErrorArgs" on page 65 • "LoginPageErrorCode" on page 66 • "PasswordExpirationFriendlyTime" on page 81

PasswordHistoryLength

	The length of the list of passwords the system has saved for a given end-user.
Object Type	Template variable
Included In	PasswordChange.thtml
Usage	Optional. Argument for error code 2008 . Some authentication systems save passwords and do not allow an end-user to assign the same password twice in succession, within a certain amount of elapsed time, or as long as the password still exists in the password history list.
Example	The following code shows the use of PasswordHistoryLength: <pre><% IF PageErrorCode == 2008 %> New password must not repeat previous <% PageErrorArgs.PasswordHistoryLength %> passwords.</pre>
Related Topics	"PasswordComplexityMessage" on page 80

pleasewait

Default text printed to the page when either ActiveX or an applet is loading.

Object Type	Template variable
Included In	Remediate.shtml
Usage	Required
Example	The following code shows a table containing two directives. The first, the loading directive, tests whether or not ActiveX or an applet is loading. If yes, then the pleasewait directive displays a message, prompting the user to wait. <pre><% loading %> <% pleasewait %></pre>
Related Topics	• “loading” on page 63 • “Remediate.shtml” on page 89

PleaseWait.shtml

	Displays a page asking the user to wait while application components install or uninstall.
Object Type	Template
Usage	You may choose to customize this template if you have configured Host Checker at the realm level in an agentless access deployment. Use this page to control Host Checker starts during pre-authentication and post-authentication. This template contains JavaScript for detecting, stopping, installing, and uninstalling the Host Checker and Cache Cleaner components as well as images and text to display to users while the performs these actions. This template also contains JavaScript that periodically checks the user’s status to determine if Host Checker or Cache Cleaner is loaded on his system and redirects him to the sign-in page (welcome.cgi) if necessary. This template also includes variables telling users to wait while components install, an error message variable, a variable storing the time that the page is first loaded, and a status variable for the Host Checker and Cache Cleaner components. This template also includes variables telling users to wait while components install, an error message variable, a variable storing the time that the page is first loaded, and a status variable for Host Checker components. For detailed information about the JavaScript and variables described here, see comments in the template.
Required Variables	• “LoginPageErrorMessage” on page 69 • “PleaseWaitObjectEC” on page 84
Example	None

Related Topics • “Logout.thtml” on page 69

pleasewaitLogoutJSCode

JavaScript code for **pleasewait** logout.

Object Type Template variable

Included In Logout.thtml

Usage Required

Example The following code shows the variable passed to the page verbatim, prior to the script section of the HTML page:

```
<html>
<head>
<meta http-equiv="Content-Language">
<meta http-equiv="Content-Type" content="text/html">
<meta name=robots content="none">
<title>Instant Virtual Extranet</title>

<% pleasewaitLogoutJSCode FILTER verbatim %>
<%IF pleasewaitObjectHC OR pleasewaitObjectCC OR pleasewaitWin32%><%IF
pleasewaitObjectHC OR pleasewaitWin32%>
<SCRIPT LANGUAGE="JavaScript">
```

Related Topics • “Logout.thtml” on page 69
• “PleaseWaitObjectEC” on page 84
• “pleasewaitWin32” on page 85

PleaseWaitObjectEC

If an Odyssey Access Client object is enabled, this variable contains the Odyssey Access Client object tag.

Object Type Template variable

Included In PleaseWait.thtml

Usage Required

Example The following code checks to see if the Odyssey Access Client object is enabled. If so, the code displays a message:

```
<%IF pleasewaitObjectEP %>
```



```
<span><div id="cc">Odyssey Access Client</div></span>
<%END%>
```

- Related Topics**
- “PleaseWait.thtml” on page 83

pleasewaitWin32

Boolean indicating that you must stop one or more Win32 components from the **logout** page.

Object Type Template variable

Included In Logout.thtml

Usage Required. Set to 1 if any components need to be stopped.

Example The following code checks to see if any of the components are causing the Infranet Controller to display a Please Wait message and if the components are java-based. If yes, the code calls the **stopComponents()** JavaScript function to stop the component.

```
<% IF (pleasewaitObjectCC || pleasewaitObjectHC || <% IF (pleasewaitObjectHC ||
pleasewaitWin32) &&
    delivery_mode == 'java'%>
    onload="javascript:stopComponents()"
<% END %>
```

- Related Topics**
- “PleaseWait.thtml” on page 83
 - “stopComponents()” on page 109

plugintype

Provides the information about type of defender server application.

Object Type Template variable

Included In Defender.thtml

Usage Optional. Valid value is CASQUEAPP. The Infranet Controller sets the plugintype variable to CASQUEAPP when finding a Casque RADIUS server.

Example The following code loads the **frmCasque** HTML form if the **plugintype** contains the value **CASQUEAPP**:

```
<% IF plugintype == 'CASQUE' %>
    <form name="frmCasque" method="POST" autocomplete=off>
    <INPUT type=hidden name="cval" value="<% secid_challenge %>">
    <INPUT type=hidden name="p" value="casqueapp">
    <a href="#" onClick="submitFrmCasque(); return false;">
```

```
    LAUNCH CASQUE PLAYER</a><BR>
</form>
```

- Related Topics**
- “Defender.shtml” on page 47
 - “frmCasque” on page 50

portal

User interface string that displays the sign-in page’s portal name.

Object Type Template variable

Default Value Secure SSL VPN

Included In LoginPage.shtml, LogoutPage.shtml, SSL.shtml, Logout.shtml, Defender.shtml, ExceededConcurrent.shtml, GeneratePin.shtml, NewPin.shtml, NextToken.shtml, ShowSystemPin.shtml

Usage Optional

Example <% portal FILTER verbatim %>

Related Topics None

prompts

Contains display information for username/password prompts.

Object Type Template variable

Included In LoginPage.shtml, NewPin.shtml, NextToken.shtml, Defender.shtml, SM- NewPinSelect.shtml, SM-NewUserPin.shtml, SM-NextToken.shtml

Usage The **prompt** variable is a hash list used to display username and password prompts in the sign-in page. You can use the **prompts** variable instead of hardcoding username and password labels.

Example The following example is a required piece of code in the **SecondaryLoginPage.shtml** template:

```
<% FOREACH prompt = prompts %>
<% NEXT IF !prompt.required %>
  <% prompt.promptText %>
  <input type="<% prompt.type %>" name="<% prompt.name %>"
    size="20">
<% END %>
```

Related Topics None

realm

Returns name of the current user's realm.

Object Type HTML form field

Included In LoginPage.thtml, SM-NewPinSelect.thtml, SM-NewPinSystem.thtml, SM-NewUserPin.thtml, SM-NextToken.thtml, GeneratePin.thtml, NewPin.thtml

Usage Required. Login.cgi passes the realm value posted here to the appropriate authentication server. See example below.

Example The following example appears in several template files:

```
<% IF RealmList.size == 0 %>
<td>LoginRealm</td><td> </td><td>
<input type="text" name="realm" size="20">
</td>
<% ELSIF RealmList.size == 1 %>
<input type="hidden" name="realm" value="<% RealmList.0 %>">
<% ELSE %>
<td>LoginRealm</td><td> </td><td>
<select size="1" name="realm">
  <% FOREACH r = RealmList %>
    <option value="<% r %>" ><% r %></option>
  <% END %>
</select>
```

If you only associate one realm with a URL, and you want to eliminate the realm text box and realm list box altogether, you can eliminate this code with the exception of the following line:

```
<input type="hidden" name="realm" value="<% RealmList.0 %>">
```

You must include this line. The Infranet Controller populates this variable with the correct value. You can use this approach if you are sure that only one realm is associated with the sign-in URL.

Related Topics "RealmList" on page 88

realm

User interface label for the string "realm".

Object Type Template variable

Default value Realm

Included In LoginPage.thtml

Usage	Optional
Example	In the following example, if no list of realms is available, display a textbox for the user to enter their realm. The realm variable is a label for the textbox. <pre><% IF RealmList.size == 0 %> <% realm %> <input type="text" name="realm" value="" size="20"></pre>
Related Topics	"RealmList" on page 88

RealmList

	List of realms available to the user.
Object Type	Template variable
Included In	LoginPage.thtml,GeneratePin.thtml, NewPin.thtml
Usage	Optional. If you use the RealmList variable in the LoginPage.thtml template, the following rules apply: • If RealmList.size = 0, then allow the user to type the realm name into a text box. • If RealmList.size => 1, then allow the user to choose the realm name from a list box.
Example	The following example from LoginPage.thtml shows RealmList being used as an index in a foreach loop that returns the available realms. <pre><% FOREACH r = RealmList %> <option value="<% r %>" ><% r %></option> <% END %></pre>
Related Topics	• "CertificateAuthentication" on page 45 • "realm" on page 87

recallLastRealmUsed()

	Use this function if you want to allow users to select from multiple realms in the sign-in page.
Object Type	JavaScript function
Included In	LoginPage.thtml
Usage	Optional
Example	The function checks that a realm exists and then gets the last selected authentication realm.

```

function recallLastRealmUsed() {
  if (document.frmLogin.realm != null &&
      document.frmLogin.realm.type == "select-one") {
    // try to remember which auth realm was last used
    var sLastRealm = GetCookieValue("lastRealm");
    if (sLastRealm.length > 0) {
      var cmb = document.frmLogin.realm;
      var nNumRealms = cmb.options.length;
      for (var n=0; n < nNumRealms; n++) {
        if (cmb.options[n].text == sLastRealm) {
          cmb.selectedIndex = n;
        }
      }
    }
  }
}

```

Related Topics "FinishLoad()" on page 49

Remediate.html

Provides the user with the ability to fix the Host Checker starts during pre-authentication and post-authentication, before the Welcome page displays.

Object Type Template

Usage Optional. If you have enabled Host Checker remediation, this page appears before/after the sign-in page and displays messages to the user when Host Checker policies with custom instructions have failed.

- Required Variables**
- "listFailedPolicies" on page 62
 - "showButtons" on page 100
 - "showClose" on page 101
 - "showContinue" on page 101
 - "showHeading" on page 102
 - "showLoading" on page 102
 - "showPolicies" on page 103
 - "showRemedOption" on page 102
 - "showTryAgain" on page 103

Example None

Related Topics None

RoleList

List of roles available to the user.

Object Type Template variable

Included In SelectRole.thtml

Usage Optional

Example The following code uses the **RoleList** variable to process a loop of roles (in this case, two roles):

```
<%FOREACH role = RoleList %>
  <a href="login.cgi?loginmode=mode_selectedrole&role=<%role.0%>
    &selectrolekey=<%SelectRoleKey%>&passwordExpiration=
    <%PasswordExpiration%>"><%role.1%></a>
<%END%>
```

Related Topics "SelectRole.thtml" on page 95

runOnLoad

Called when page loads the **<body>**.

Object Type JavaScript function

Included In Remediate.thtml

Usage Optional. Useful when you need to load ActiveX or an applet as needed.

Example None

Related Topics "Remediate.thtml" on page 89

safari

Boolean that indicates whether or not the user's browser is Macintosh Safari.

Object Type Template variable

Included In Logout.thtml

Usage Required

Example The following code fragment tests the variable to discover if the browser is the Safari browser:

`<% IF safari %>`

Related Topics “geckoBrowser” on page 53

secid_challenge

Challenge string obtained from a server.

Object Type Template variable

Included In Defender.thtml

Usage Required. Used to return a challenge string from a RADIUS server in **Defender.thtml**.

Example The following code sets the challenge value to the value obtained from the RADIUS server.

```
<INPUT type=hidden name="cval" value="<% secid_challenge %>">
```

Related Topics

- “Defender.thtml” on page 47
- “p” on page 76

secidcontextid

Form field for the value represented by **secid_contextid**.

Object Type HTML form field

Included In GeneratePin.thtml, NewPin.thtml, NextToken.thtml, Defender.thtml

Usage Required

Example The following code assigns the handle of the current transaction to a form field:

```
<INPUT type=hidden name="secidcontextid" value="<% secid_contextid %>">
```

Related Topics “secid_contextid” on page 91

secid_contextid

Current transaction handle.

Object Type Template variable

Included In GeneratePin.thtml, NewPin.thtml, NextToken.thtml

Usage	Required. You must send this value to the server, but if you do not want to expose it to the user, send it as a hidden form field.
Example	The following code assigns the current transaction handle from the server to the HTML form. <code><INPUT type=hidden name="secidcontextid" value="<% secid_contextid %>"></code>
Related Topics	"secidcontextid" on page 91

secid_loginmode

	Returns a value of Next Token mode or New PIN mode.
Object Type	Template variable
Included In	GeneratePin.thtml, NewPin.thtml, NextToken.thtml, Defender.thtml
Usage	Required. You must send this value to the server, but if you do not want to expose it to the user, you may send it as a hidden form field.
Example	The following code assigns the mode to the HTML form: <code><INPUT type=hidden name="loginmode" value="<% secid_loginmode %>"></code>
Related Topics	• "secidcontextid" on page 91 • "secid_contextid" on page 91

secid_pinerr

	Error code and corresponding message that informs the user if he has entered his PIN incorrectly.
Object Type	Template variable
Included In	GeneratePin.thtml, NewPin.thtml
Usage	Optional
Example	The following code shows four different conditions using secid_pinerr :

```
<% IF secid_pinerr == 0 %>
  New PIN required
<% ELSIF secid_pinerr == 1 %>
  The Two PINs Entered Do Not Match
<% ELSIF secid_pinerr == 2 %>
  Invalid PIN Format
<% ELSIF ecid_pinerr == 3 %>
```


Invalid PIN Length
 <% END %>

- Related Topics**
- “secid_pinformat” on page 93
 - “secid_pinselectmode” on page 93

secid_pinformat

Error message informing the end-user of the restrictions on the required PIN.

Object Type Template variable

Included In GeneratePin.thtml, NewPin.thtml

Usage Optional

Example The following fragment shows how to use this variable to display a particular restriction:

You must create a new Personal Identification Number (PIN) before you can sign in. Your PIN should be <% secid_pinformat %> long.

Related Topics “secidpinformat” on page 94

secid_pinselectmode

Sets the user's PIN select mode.

Object Type Template variable

Included In GeneratePin.thtml, NewPin.thtml

Usage Optional

Example The following code creates a link and sets the login mode:

```
<% IF secid_pinselectmode == 1 %>
  If you prefer, the system can
  <a href="secid_genpin.cgi?loginmode=<% secid_loginmode %>"
    onclick="gCancelNewPinMode=false;">
    generate a PIN</a> for you.
<% END %>
```

Related Topics “secidpinselectmode” on page 95

secid_systempin

System-generated PIN.

Object Type Template variable

Included In	ShowSystemPin.shtml,GenerateNewPin.shtml, NewPin.shtml
Usage	Optional
Example	The following code sets the user's PIN. Your new PIN is <% secid_systempin %> Be sure to remember it, because you need your PIN each time you sign in.
Related Topics	"secidactionSavePin" on page 94

secid_username

	System-supplied username.
Object Type	Template variable
Included In	GeneratePin.shtml, NewPin.shtml, NextToken.shtml, Defender.shtml, LoginPage.shtml
Usage	Required
Example	The following code posts the username passed into the form back to the server: <INPUT type=hidden name="username" value="<% secid_username %>">
Related Topics	

secidactionSavePin

	Specifies an action to be taken when a new PIN is created.
Object Type	HTML form field
Included In	NewPin.shtml,GeneratePin.shtml
Usage	Required
Example	The value of one of the following fields must be sent to the Infranet Controller server: <INPUT type="submit" name="secidactionSavePin" value="Save PIN"> <INPUT type="submit" name="secidactionCancel" value="Cancel">
Related Topics	• "GeneratePin.shtml" on page 54 • "NewPin.shtml" on page 75

secidpinformat

Error message informing the end-user of the restrictions on the required PIN.

Object Type HTML form field

Included In NewPin.thtml,GeneratePin.thtml

Usage Required

Example The following code assigns the secure ID PIN format:

```
<INPUT type=hidden name="secidpinformat" value="<% secid_pinformat %>">
```

Related Topics "secid_pinformat" on page 93

secidpinselectmode

Passes the user's PIN select mode to the server.

Object Type HTML form field

Included In NewPin.thtml, GeneratePin.thtml

Usage Required

Example The following code assigns the PIN select mode:

```
<INPUT type=hidden name="secidpinselectmode" value="<% secid_pinselectmode %>">
```

Related Topics "secid_pinselectmode" on page 93

SelectRole.thtml

Appears after the sign-in page and displays a list of roles from which the user can choose.

Object Type Template

Usage Optional. You may choose to customize this template if you have assigned your users to multiple roles, but have not permissively merged those roles.

Required Variables None

Example None

Related Topics None

setFailed()

Called if the component is stopped intentionally.

Object Type	JavaScript function
Included In	Logout.thtml, PleaseWait.thtml
Usage	component signifies the component that is being checked to see if it has been stopped intentionally.
Example	The following code stops Host Checker: <pre><%IF pleasewaitObjectHC %> setSucceeded('hc'); <%END%> }catch (e) { <%IF pleasewaitObjectHC %> setFailed('hc'); <%END%> }</pre>
Related Topics	"setStarted()" on page 97

setFinished

	Called if the component has finished starting up.
Object Type	JavaScript function
Included In	Logout.thtml, PleaseWait.thtml
Usage	Optional <i>component</i> —Valid value for component is hc for Host Checker.
Example	The following code shows how setFinished is used in the Logout.thtml template: <pre>function setFinished(component) { document.getElementById(component).style.fontWeight = "normal"; }</pre>
Related Topics	• "Logout.thtml" on page 69 • "PleaseWait.thtml" on page 83

SetLastRealm()

	Enables users to select from multiple realms on the sign-in page.
Object Type	JavaScript function
Included In	LoginPage.thtml

Usage When the user signs in, this function retrieves the authentication realm selected by the user and sets an expiration date of 30 days.

sValue—The name of the realm the user selects.

Example The following code shows the function:

```
function SetLastRealm(sValue) {
  var dtExpire = new Date();
  dtExpire.setDate(dtExpire.getDate() + 30);
  document.cookie = "lastRealm=" +
    escape(sValue) + "; expires=" + dtExpire.toGMTString();
}
```

Related Topics “LoginPage.thtml” on page 65

setStarted()

Called if the component is started in order to stop it.

Object Type JavaScript function

Included In Logout.thtml, PleaseWait.thtml

Usage Required.

component—Name of the component.

Example The following code shows the function:

```
function setStarted(component)
{
  document.getElementById(component).style.fontWeight = "bold";
}
```

Related Topics “setFailed()” on page 95

setSucceeded()

Called if the component is successfully stopped.

Object Type JavaScript function

Included In Logout.thtml, PleaseWait.thtml

Usage Required

Example The following code calls the function if Host Checker has been successfully stopped:

```
<%IF pleasewaitObjectHC %> setSucceeded('hc'); <%END%>
```

Related Topics “setStarted()” on page 97

setthankyou()

Sets the timeout value and calls the **thankyou()** function, which displays the login page after the user has signed out.

Object Type JavaScript function

Included In Logout.thtml

Usage Required

Example The following code shows the function:

```
function setthankyou()
{
  setTimeout("thankyou()", 2000);
}
```

Related Topics “thankyou()” on page 112

setup_classid

The CLASSID of the ActiveX object.

Object Type Template variable

Included In Logout.thtml, PleaseWait.thtml

Usage Required

Example The following example assigns the CLASSID if the delivery mechanism is ActiveX.

```
<%IF delivery_mode == 'activex'%>
<OBJECT <%setup_classid FILTER verbatim%> id=NeoterisSetup
```

Related Topics • “setup_codebase” on page 98

setup_codebase

The path and version information (CODEBASE) of the ActiveX object.

Object Type Template variable

Included In Logout.thtml, PleaseWait.thtml

Usage Required

Example The following example assigns the CODEBASE if the delivery mechanism is ActiveX.

```
<%IF delivery_mode == 'activex'%>
  <OBJECT <%setup_classid FILTER verbatim%> id=NeoterisSetup
  <%setup_codebase FILTER verbatim%>
  width=0 height=0 >
</OBJECT>
```

Related Topics • “setup_classid” on page 98

Setupappversion

The version of the Java installer setup application.

Object Type Template variable

Included In Logout.thtml

Usage Required

Example The following example assigns the **setupappversion** after determining the delivery mode:

```
<% IF delivery_mode == 'java' %>
  <APPLET code=dsSetupApplet.class
  code=dsSetupApplet
  id=NeoterisSetup
  archive="NeoterisSetupApplet.jar"
  MAYSCRIPT
  width=0 height=0
  codebase="/dana-cached/setup/" >
  <PARAM NAME="SetupAppVersion" VALUE="<% setupappversion %>">
```

Related Topics “Logout.thtml” on page 69

setup_codebase

Codebase of Juniper Networks NeoterisSetup ActiveX control.

Object Type Template variable

Included In Logout.thtml, PleaseWait.thtml

Usage Optional

Example The following code declares the **codebase**:

```
<OBJECT classid="clsid:4CC35DAD-40EA-4640-ACC2-A1A3B6FB3E06"
id=NeoterisSetup
<% setup_codebase FILTER verbatim %>
```

```
width=0 height=0 >
</OBJECT>
```

- Related Topics**
- “Logout.thtml” on page 69
 - “PleaseWait.thtml” on page 83

showButtons

Boolean indicating whether or not to show any of the action buttons.

Object Type Template variable

Included In Remediate.thtml

Usage Required

Example The following code tests **showButtons**. If true, then the code continues to test other directives to determine which action buttons to display:

```
<% IF showButtons %>
  <% IF showTryAgain %>
    <input name="btnTryAgain" type="button" value="<% textTryAgain %>"
      onClick="TryAgain()" <% disabled %>>
  <% END %>
  <% IF showRemedOption %>
    <input name="btnHideRemed" type="button"
      value="<% textRemedOption %>" onClick="HideRemed();
    <% disabled %>>
    <% HC_REMED_SHOW_ADV_PREFS %>
  <% END %>
  <% IF showContinue %>
    <input name="btnContinue" type="button" value="<% textContinue %>"
      onClick="Continue()">
  <% END %>
  <% IF showClose %>
    <input name="btnClose" type="button" value=" <% textClose %> "
      onClick="javascript:self.close()">
  <% END %>
<% END %>
```

- Related Topics**
- “showContinue” on page 101
 - “showClose” on page 101
 - “showRemedOption” on page 102
 - “showTryAgain” on page 103

showChangePasswordTitle

Boolean indicating whether or not to show password change title.

Object Type Template variable

Included In	PasswordChange.shtml
Usage	Optional.
Example	The following code tests the variable. If true, the code then displays the value contained in <code>changePasswordTitle</code> :
	<pre><%IF showChangePasswordTitle %> <% changePasswordTitle FILTER verbatim %> <%END%></pre>
Related Topics	• “changePasswordTitle” on page 45 • “PasswordChange.shtml” on page 80

showClose

	Boolean indicating whether or not to show the Close button.
Object Type	Template variable
Included In	Remediate.shtml
Usage	Required
Example	The following code tests showClose . If true, the code constructs a button that closes the page when the user clicks it:
	<pre><% IF showClose %> <input name="btnClose" type="button" value=" <% textClose %> " onClick="javascript:self.close()"> <% END %></pre>
Related Topics	“showContinue” on page 101

showContinue

	Boolean indicates whether or not to show the Continue button.
Object Type	Template variable
Included In	Remediate.shtml
Usage	Required
Example	The following code shows a Continue button if the showContinue variable is true:
	<pre><% IF showContinue %> <input name="btnContinue" type="button" value="<% textContinue %>"</pre>

```
onClick="Continue()"></td>
<% END %>
```

Related Topics "Remediate.thtml" on page 89

showHeading

flag to determine if we should show heading

Object Type Template variable

Included In Remediate.thtml

Usage Required

Example The following code displays message text if the **showHeading** variable is true:

```
<% IF showHeading %>
  <% msg FILTER verbatim %>
<% END %>
```

Related Topics "Remediate.thtml" on page 89

showLoading

Boolean indicates whether or not to show text during the loading of ActiveX or an applet.

Object Type Template variable

Included In Remediate.thtml

Usage Required

Example The following code displays message text if the **showLoading** variable is true:

```
<% IF showLoading %>
  <% pleasewait %>
  <% HC_REMED_POLICIES_CHECK %>
```

Related Topics "Remediate.thtml" on page 89

showRemedOption

Boolean indicates whether or not to show the button that displays the message: Do not show remediation for this session.

Object Type Template variable

Included In Remediate.thtml

Usage Required

Example The following code tests the value of `showRemedOption`. If true, the code hides the remediation button:

```
<% IF showRemedOption %>
  <input name="btnHideRemed" type="button"
    value="<% textRemedOption %>" onClick="HideRemed(); <% disabled %>">
  <% HC_REMED_SHOW_ADV_PREFS %>
<% END %>
```

Related Topics “Remediate.thtml” on page 89

showPolicies

Boolean indicating whether or not to show remediation policies.

Object Type Template variable

Included In Remediate.thtml

Usage Required

Example The following code tests the values of several template variables:

```
<% IF showHeading || showPolicies || showButtons %>
```

Related Topics “Remediate.thtml” on page 89

ShowSystemPin.thtml

Displays the system-generated PIN to the user.

Object Type Template

Usage Optional

Required Variables None

Example None

Related Topics None

showTryAgain

Boolean that indicates whether or not to show the **Try Again** button.

Object Type Template variable

Included In	Remediate.shtml
Usage	Required
Example	The following code checks the variable and, if true, displays the Try Again button. The code also disables the button once the end-user has clicked it: <pre><% IF showTryAgain %> <input name="btnTryAgain" type="button" value="<% textTryAgain %>" onClick="TryAgain()" <% disabled %>> <% END %></pre>
Related Topics	"TryAgain" on page 113

signinAgain

	Boolean that indicates whether or not to display the login page after users log out.
Object Type	Template variable
Included In	Logout.shtml
Usage	Optional
Example	The following code checks the variable and, if true, redirects the user to the sign-in page (welcome.cgi). <pre><% IF signinAgain %> <div id=signindiv> <% signinAgain FILTER verbatim %></pre>
Related Topics	None

signInLink

	Provides the link to the start page.
Object Type	Form variable
Included In	Logout.shtml
Usage	Optional. If you want to provide the user the ability to sign in again, you must include this link in the href tag.
Example	None
Related Topics	"Logout.shtml" on page 69

SM-NewPinSelect.shtml

Prompts the user to enter a new PIN or create a system-generated PIN before signing into the Infranet Controller.

Object Type Template

Usage Required for SiteMinder authentication.

Required Variables "secid_loginmode" on page 92

Example None

Related Topics None

SM-NewPinSystem.shtml

Enables the user to create a system-generated PIN if the user selects the **System PIN** option in the **SM-NewPinSelect.shtml** page.

Object Type Template

Usage Optional

Required Variables "secid_loginmode" on page 92

Example None

Related Topics "SM-NewPinSelect.shtml" on page 105

SM-NewUserPin.shtml

Prompts the user to create a new PIN if he selects the **Enter PIN** option in the **SM-NewPinSelect.shtml** page.

Object Type Template

Usage Optional. Also determines if the two PINs entered by the user match and alerts the user if necessary. When configuring the **secid_pinserr** variable (which stores the error code and message that informs the user if he mis-entered his PIN), note that possible values include:

Table 12: New PIN Assignment Messages

Code	Value
0	New PIN Assignment

Table 12: New PIN Assignment Messages (*continued*)

Code	Value
1	The Two PINs Entered Do Not Match

Required Variables “secid_loginmode” on page 92

Example None

Related Topics “SM-NewPinSelect.shtml” on page 105

SM-NextToken.shtml

Prompts the user to verify his credentials by entering his SecureID token code.

Object Type Template

Usage Optional. For information the JavaScript, form definitions, form fields, and variables contained in these templates, see comments in the templates.



NOTE: If you prohibit the use of System-generated PINS, you do not need to include SM-NewPinSelect.shtml and SM-NewPinSystem.shtml in the zip file that you upload to the Infranet Controller.

Required Variables “secid_loginmode” on page 92

Example None

Related Topics

- “SM-NewPinSelect.shtml” on page 105
- “SM-NewPinSystem.shtml” on page 105

softid_error

Set if the ACE/SoftID authentication server returns errors.

Object Type Template variable

Included In GeneratePin.shtml (ACE/softID), LoginPage.shtml (ACE/softID), NewPin.shtml (ACE/softID), NextToken.shtml (ACE/softID)

Usage Required

Example The following code assigns any SoftID errors to a JavaScript variable:

```
var error= "<% softid_error %>";
```

Related Topics “softid_time” on page 107

softid_time

Contains the ACE/SoftID authentication server time.

Object Type Template variable

Included In LoginPage.shtml (ACE/softID)

Usage Required by **authenticate()** function.

Example The following code assigns the ACE/SoftID time to a time variable:

```
var time = <% softid_time %> ;
```

Related-topics “LoginPage.shtml” on page 65

SSL.shtml

Displays an error if authentication fails and the user is not allowed to sign into the Infranet Controller.

Object Type Template

Usage You must always include **SSL.shtml** in your zip file. This is a standard Infranet Controller page that displays an error if authentication fails or realm-level checks fail and the user is not allowed to sign into the Infranet Controller. This template does not contain any JavaScript or forms. However, it does contain error message variables. For more information, see comments in the template.

Required Variables None

Example None

Related Topics None

start_status

Status of the started Host Checker or Cache Cleaner instance. Status of the started Host Checker instance.

Object Type JavaScript variable

Included In PleaseWait.shtml

Usage Required. The return status equals 1 if a test of this variable fails.

Example The following code shows how the `getStatus` function uses the `start_status` variable:

```
function getStatus()  
{  
  return start_status;  
}
```

Related Topics “PleaseWait.thtml” on page 83

- `StartCC()` Enables the current user to start Cache Cleaner. Object Type JavaScript function Included In Usage Optional Example None Related Topics “StartEP()” on page 108

StartEP()

Enables the current user to start the Odyssey Access Client.

Object Type JavaScript function

Included In `AgentInstall.thtml`

Usage Optional

Example None

Related Topics • “PleaseWait.thtml” on page 83

StartHC()

Enables the current user to start Host Checker.

Object Type JavaScript function

Included In `PleaseWait.thtml`

Usage Optional

Example None

Related Topics • “PleaseWait.thtml” on page 83

startPageLink

Page Link for the Welcome Page.

Object Type Template variable

Included In GraceLoginUsed.thtml, PasswordExpiration.thtml

Usage Required. Must be added as an href link. Provides a link in the page, allowing user to navigate to the Welcome page.

Example The following code shows this variable used as an href link:

```
<a href="<% startPageLink %>">Click here to go to Welcome Page</a>
```

- Related Topics**
- “GraceLoginUsed.thtml” on page 55
 - “PasswordExpiration.thtml” on page 81

stopComponents()

Performs a variety of shutdown checks and operations, shutting down Host Checker, Cache Cleaner, and Win32 applets, if necessary, when the end-user signs out.

Object Type JavaScript function

Included In Logout.thtml

Usage Required

Example

```
<% IF (pleasewaitObjectCC || pleasewaitObjectHC || pleasewaitWin32) && <% IF
(pleasewaitObjectHC || pleasewaitWin32) &&
  delivery_mode == 'java'%>
  onload="javascript:stopComponents()"
<%END%>
```

- Related Topics** “delivery_mode” on page 47

SubmitClicked()

Saves or cancels a PIN per the user's request.

Object Type JavaScript function

Included In GeneratePin.thtml, NewPin.thtml, NextToken.thtml, SM-NewUserPin.thtml

Usage Required. You must include this function in the header of GeneratePin.thtml.

Example From GeneratePin.thtml:

```
function SubmitClicked() {
  gCancelNewPinMode = false;
  return true;
}
```

From SM-NewUserPin.thtml:

```
function SubmitClicked() {  
    if (document.frmNewPin.password2.value !=  
        document.frmNewPin.password3.value) {  
        alert("The Two PINs Entered Do Not Match");  
        document.frmNewPin.password2.focus();  
        return false;  
    }  
}
```

- Related Topics**
- “GeneratePin.shtml” on page 54
 - “NewPin.shtml” on page 75
 - “NextToken.shtml” on page 76
 - “SM-NewUserPin.shtml” on page 105

submitFrmCasque()

Initiates the download of the CASQUE RADIUS server.

Object Type JavaScript function

Included In Defender.shtml

Usage Required if CASQUE RADIUS server is needed.

Example The following code shows the action to initiate the CASQUE server:

```
function submitFrmCasque() {  
    document.frmCasque.action = "/dana-na/download/x.casque?  
    url=/dana-na/auth/welcome.cgi";  
    document.frmCasque.submit();  
}
```

- Related Topics**
- “Defender.shtml” on page 47
 - “frmCasque” on page 50

textClose

Default text for the **Close** button.

Object Type Template variable

Included In Remediate.shtml

Usage Required. Used by the Infranet Controller to dynamically update the button name, depending upon the user’s activity.

Example The following code changes the button to a **Close** button:

```
<% IF showClose %>
<input name="btnClose" type="button" value=" <% textClose %> "
  onClick="javascript:self.close()">
<% END %>
```

- Related Topics**
- “textContinue” on page 111
 - “textRemedOption” on page 111
 - “textTryAgain” on page 112

textContinue

Default text for the **Continue** button.

Object Type Template variable

Included In Remediate.thtml

Usage Required. Used by the Infranet Controller to dynamically update the button name, depending upon the user’s activity.

Example

```
<% IF showContinue %>
<input name="btnContinue" type="button" value="<% textContinue %>"
  onClick="Continue()">
<% END %>
```

- Related Topics**
- “textClose” on page 110
 - “textRemedOption” on page 111
 - “textTryAgain” on page 112

textRemedOption

Default text for the **Do not show remediation for this session** button.

Object Type Template variable

Included In Remediate.thtml

Usage Required. Used by the Infranet Controller to dynamically update the button name, depending upon the user’s activity.

Example

```
<% IF showRemedOption %>
<input name="btnHideRemed" type="button"
  value="<% textRemedOption %>"
  onClick="HideRemed(); <% disabled %>">
```

```
<% HC_REMED_SHOW_ADV_PREFS %>
<% END %>
```

- Related Topics**
- “textClose” on page 110
 - “textContinue” on page 111
 - “textTryAgain” on page 112

textTryAgain

Default text for the **Try Again** button.

Object Type Template variable

Included In Remediate.thtml

Usage Required. Used by the Infranet Controller to dynamically update the button name, depending upon the user’s activity.

Example

```
<% IF showTryAgain %>
<input name="btnTryAgain" type="button" value="<% textTryAgain %>"
  onClick="TryAgain()" <% disabled %>>
<% END %>
```

- Related Topics**
- “textClose” on page 110
 - “textContinue” on page 111
 - “textRemedOption” on page 111

thankyou()

Removes components and displays the login page after an end-user signs out.

Object Type JavaScript function

Included In Logout.thtml

Usage Optional

Example The function checks to see which application is being stopped and removed, then displays either an error or the sign-in page.

```
function thankyou()
{
<%IF pleasewaitObjectHC %>
  removeBulb('hc');
  removeComponent('hc');
<%END%>
<%IF pleasewaitObjectCC %><%END%>
```

```
document.getElementById('pleasewait1').innerHTML =
"<% LoginPageErrorMessage FILTER verbatim%>";
document.getElementById('signindiv').innerHTML =
"Click here to sign in again";
}
```

Related Topics “setthankyou()” on page 98

TryAgain

Submits the hidden form when the user clicks the **Try Again** button.

Object Type JavaScript function

Included In Remediate.thtml

Usage Required

Example

```
<% IF showTryAgain %>
<input name="btnTryAgain" type="button" value="<% textTryAgain %>"
  onClick="TryAgain()" <% disabled %>>
<% END %>
```

Related Topics “showTryAgain” on page 103

tz_offset

Contains the time zone offset.

Object Type HTML form field

Included In LoginPage.thtml

Usage Required. The **Login()** function uses the **tz_offset** (time zone offset) value to help determine the user’s time zone.

Example The following code shows how the **tz_offset** field is used in an HTML form.

```
<input type="hidden" name="tz_offset">
```

Related Topics “LoginPage.thtml” on page 65

upAndRunning()

Indicates whether or not Host Checker or Cache Cleaner are up and running after they are started. Indicates whether or not Host Checker is up and running after it is started.

Object Type JavaScript function

Included In	PleaseWait.shtml
Usage	Optional
Example	None
Related Topics	"PleaseWait.shtml" on page 83

welcome

	Contains the welcome message string.
Object Type	Template variable
Default Value	"Welcome to the"
Included In	LoginPage.shtml, LogoutPage.shtml, SSL.shtml, Logout.shtml, Defender.shtml, ExceededConcurrent.shtml, GeneratePin.shtml, NewPin.shtml, NextToken.shtml, ShowSystemPin.shtml
Usage	Optional
Example	<pre><form name="frmLogin" action=login.cgi method="POST" autocomplete=off onsubmit="return Login(<% setcookies %>)"> <input type="hidden" name="tz_offset"> <% welcome FILTER verbatim %></pre>
Related Topics	None

win32

	Indicates whether or not the client operating system is Windows.
Object Type	Template variable
Included In	Logout.shtml
Usage	Required
Example	The following code checks to see if the operating system is Windows. If yes, the code loads redirects the page to a page to test the compatibility of Microsoft Java: <pre>function checkActiveX () { var win = <% win32 %>; try { NeoterisSetup.isValid(); document.form1.submit(); return true; }</pre>

```
} catch (e) {  
  var url = win ? "/dana-na/meeting/  
    <%signin%>meeting_testmsjava.cgi?  
    <%mid_param%>": "/dana-na/meeting/<%signin%>  
    meeting_testjava.cgi?<%mid_param%>";  
  document.location.replace(url);  
}  
}
```

Related Topics None

PART 2

Index

- Index on page 119

Index

A

Accept-Language.....	36
Accept-Language header abbreviations.....	37
ActivePerl.....	11
AnonymousAuthentication variable, discussed.....	41
AppConfig.....	11
arithmetic operators, defined.....	19
authentication servers	
SiteMinder	
custom sign in pages.....	49

B

block directives, defined.....	20
--------------------------------	----

C

CALL directive, restriction.....	21
Cancel.shtml, discussed.....	45
CASE directive, discussed.....	20
CertificateAuthentication variable, discussed.....	45
conditional operators, discussed.....	19
conditional test, creating.....	20
custom help files.....	10
custom sign-in pages.....	36
localizing.....	36
customer support.....	xviii
contacting JTAC.....	xviii
Cygwin.....	11

D

Defender.shtml, discussed.....	47
directive, defined.....	13

E

ELSIF directives, discussed.....	20
END directive, discussed.....	20
errors	
modifying.....	66
ExceededConcurrent.shtml, discussed.....	48

F

FILTER directive, restriction.....	21
FinishLoad() function, discussed.....	48
FOREACH loop, discussed.....	21

G

GeneratePin.shtml, discussed.....	54
GET directive, discussed.....	15
GetCookieValue(sName) function, discussed.....	54
getTimezoneOffset() function, discussed.....	64
GraceLoginUsed.shtml, discussed.....	55

H

Home variable, discussed.....	59
Host Checker	
custom pages.....	69, 83

I

IF directive, discussed.....	20
INCLUDE.....	15
INSERT.....	15
internationalization.....	36
INTERPOLATE directive, restriction.....	21

K

keyboard.js.....	61
Kiosk.zip.....	8
Kiosk.zip, discussed.....	38

L

languages.....	36
Linux.....	11
localization.....	36
Login() function, discussed.....	63
LoginPage.shtml, customizing.....	65
LoginPage.shtml, discussed.....	65
LoginPage.ErrorCode.....	66
LoginPage.ErrorCode variable, discussed.....	69
LoginPage.ErrorMessage variable,	
discussed.....	66, 69

Logout.html, discussed.....69

M

Mac OS X.....11

mathematical operators, defined.....19

N

NewPin.html, discussed.....75

NextToken.html, discussed.....75

P

PasswordChange.html, discussed.....80

PasswordExpiration.html, discussed.....81

Perl AppConfig.....11

PERL directive, restriction.....21

Pocket PC

 custom sign in pages.....25

PROCESS.....14, 15

R

RAWPERL directive, restriction.....21

RealmList variable, discussed.....88

recallLastRealmUsed() function, discussed.....88

RSA Soft ID client.....8

RSA SoftID client, custom pages.....38

S

sample templates

 downloading.....12

Sample.zip.....8

samples.zip, discussed.....22

secid_pinselectmode variable, discussed.....75

secid_pinserr variable, discussed.....75

SelectRole.html, discussed.....95

SET directive, discussed.....15

SetLastRealm(sValue) function, discussed.....95

ShowSystemPin.html, discussed.....103

sign-in page, customizing.....65

SM-NewPinSelect.html, discussed.....105

SM-NewPinSystem.html, discussed.....105

SM-NewUserPin.html, discussed.....105

SM-NextToken.html, discussed.....106

SoftID.zip.....8

SoftID.zip, discussed.....38

support, technical See technical support

SWITCH directive, discussed.....20

T

TAGS directive, restriction.....21

technical support

 contacting JTAC.....xviii

template comments.....15

Template Toolkit.....10

Template Toolkit documentation.....15

template toolkit language, discussed.....13

template variable

 LoginPageErrorCode.....66

thtml files.....36

tpage.....12

ttree.....12

tz_offset variable, discussed.....113

tz_offset variable, discussed.....64

U

USE directive, restriction.....21

W

WHILE loop, discussed.....21

Windows.....11