



Juniper Networks IVE Platform
J.E.D.I. Solution Guide

Release 5.4

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

Part Number: 54A071706CDH

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.

Juniper Networks, the Juniper Networks logo, NetScreen, NetScreen Technologies, the NetScreen logo, NetScreen-Global Pro, ScreenOS, and GigaScreen are registered trademarks of Juniper Networks, Inc. in the United States and other countries.

The following are trademarks of Juniper Networks, Inc.: ERX, E-series, ESP, Instant Virtual Extranet, Internet Processor, J2300, J4300, J6300, J-Protect, J-series, J-Web, JUNOS, JUNOScope, JUNOScript, JUNOSe, M5, M7i, M10, M10i, M20, M40, M40e, M160, M320, M-series, MMD, NetScreen-5GT, NetScreen-5XP, NetScreen-5XT, NetScreen-25, NetScreen-50, NetScreen-204, NetScreen-208, NetScreen-500, NetScreen-5200, NetScreen-5400, NetScreen-IDP 10, NetScreen-IDP 100, NetScreen-IDP 500, NetScreen-Remote Security Client, NetScreen-Remote VPN Client, NetScreen-SA 1000 Series, NetScreen-SA 3000 Series, NetScreen-SA 5000 Series, NetScreen-SA Central Manager, NetScreen Secure Access, NetScreen-SM 3000, NetScreen-Security Manager, NMC-RX, SDX, Stateful Signature, T320, T640, T-series, and TX Matrix. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners. All specifications are subject to change 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.

Copyright © 2006, Juniper Networks, Inc.
All rights reserved. Printed in USA.

Juniper Networks Secure Access J.E.D.I. Solution Guide, Release 5.4
Writers: Claudette Hobbart, Paul Battaglia

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.

Year 2000 Notice

Juniper Networks hardware and software products are Year 2000 compliant. The JUNOS software has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

Software License

The terms and conditions for using this software are described in the software license contained in the acknowledgment to your purchase order or, to the extent applicable, to any reseller agreement or end-user purchase agreement executed between you and Juniper Networks. By using this software, you indicate that you understand and agree to be bound by those terms and conditions.

Generally speaking, the software license restricts the manner in which you are permitted to use the software and may contain prohibitions against certain uses. The software license may state conditions under which the license is automatically terminated. You should consult the license for further details.

For complete product documentation, please see the Juniper Networks Web site at www.juniper.net/techpubs.

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 Juniper Networks, Inc. and its subsidiaries (collectively "Juniper"), and 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, and updates and releases of such software, for which Customer has paid the applicable license or support fees to Juniper or an authorized Juniper reseller.

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:

- Customer shall use the Software solely as embedded in, and for execution on, Juniper equipment originally purchased by Customer from Juniper or an authorized Juniper reseller, unless the applicable Juniper documentation expressly permits installation on non-Juniper equipment.
- 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.
- 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, 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. Customer's use of the Software shall be subject to all such limitations and purchase of all applicable licenses.

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 the Software on non-Juniper equipment where the Juniper documentation does not expressly permit installation on non-Juniper equipment; (j) use the Software (or make it available for use) on Juniper equipment that the Customer did not originally purchase from Juniper or an authorized Juniper reseller; or (k) 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 NON-INFRINGEMENT. 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 for the Software are exclusive of taxes, withholdings, duties, or levies (collectively "Taxes"). Customer shall be responsible for paying Taxes arising from the purchase of the license, or importation or use of the Software.

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 présentés 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)).

Table of Contents

Chapter 1	J.E.D.I. overview	1
Chapter 2	Using the IVE Web console to implement endpoint solutions	3
	Task summary: Configuring Host Checker	3
	Securing managed vs. unmanaged endpoints using remediation	5
	Configuring Host Checker to secure managed endpoints	5
	Understanding Host Checker remediation	6
	Specifying remediation options	7
	Securing endpoints using custom rules in Host Checker policies	8
	Specifying custom rules in Host Checker policies	9
	Custom rule: Screen saver check	12
	Custom rule: Microsoft hotfix installation check	13
	Custom rule: Windows update status verification	13
	Custom rule: Network bridge disabled	14
	Custom rule: IP forwarding disabled	14
	Custom rule: Internet connection sharing disabled	14
	Custom rule: Google Desktop Search check	15
	Securing endpoints using the Sygate Security Agent	16
	Configuring the Sygate Management Server	16
	Configuring the Sygate Security Agent	16
Chapter 3	Using J.E.D.I. interfaces to implement endpoint solutions	17
	Choosing which interface to use	18
	Using the Host Check Client Interface	18
	Signing your custom DLL	19
	Deploying and maintaining your custom DLL	20
	NetScreen Host Checker (NHC) API	20
	Using the Host Check Server Integration Interface	21
	Deploying third party applications through Host Checker	21
	Creating your interface DLL	26

Chapter 1

J.E.D.I. overview

Juniper Networks developed the Juniper Endpoint Defense Initiative (J.E.D.I.) to provide a comprehensive solution to assess the trust worthiness of IVE endpoints. Using J.E.D.I. components, you can secure the systems of users inside and outside your network before allowing them to connect to your IVE appliance.

J.E.D.I. provides two major methods for implementing first-class endpoint defense solutions:

- **IVE Web console**—Using standard options in any IVE appliance’s Web console, you can configure Host Checker to perform endpoint checks on hosts that connect to the IVE. Host Checker checks for third party applications, files, process, ports, registry keys, and custom DLLs and denies or enables access based on the results of the checks. When properly licensed, you can also use Host Checker to download advanced malware detection software directly to the user’s computer. When a user’s computer does not meet the requirements you specify, you can display remediation instructions to users so they can bring their computers into compliance. For more information, see “Using the IVE Web console to implement endpoint solutions” on page 3.
- **J.E.D.I. APIs (Windows only)**—Using the J.E.D.I. APIs (also called Host Checker Interfaces), you create custom solutions for Windows clients that integrate third-party products into the IVE products. You can use the Host Check Client Interface to run your own DLLs through Host Checker or use the Host Check Server Integration Interface to tightly integrate your own DLLs and corresponding files into IVE appliances. For more information, see “Using J.E.D.I. interfaces to implement endpoint solutions” on page 17.

For descriptions of the Host Checker component included in J.E.D.I., see the *Juniper Networks Secure Access Administration Guide*.

Chapter 2

Using the IVE Web console to implement endpoint solutions

Host Checker is a client-side agent that performs endpoint checks on hosts that connect to the IVE. You can invoke Host Checker before displaying an IVE sign-in page to a user and when evaluating a role mapping rule or resource policy.

The IVE can check hosts for endpoint properties using a variety of rule types, including rules that check for and install advanced malware protection; predefined rules that check for antivirus software, firewalls, malware, spyware, and specific operating systems from a wide variety of industry leaders; and custom rules that check for certain third party DLLs, ports, processes, files, and registry key settings.

If the user's computer does not meet any of the Host Checker policy requirements, you can display a custom-made HTML remediation page to the user. This page can contain your specific instructions as well as links to resources to help the user bring his computer into compliance with each Host Checker policy.

This section includes the following information about Host Checker:

- “Task summary: Configuring Host Checker” on page 3
- “Securing managed vs. unmanaged endpoints using remediation” on page 5
- “Securing endpoints using custom rules in Host Checker policies” on page 9
- “Securing endpoints using the Sygate Security Agent” on page 16

Task summary: Configuring Host Checker

This section describes the high-level steps required to configure Host Checker through the IVE Web console.

For more detailed implementation instructions, see the *Juniper Networks Secure Access Administration Guide*.

To configure Host Checker, you must perform these tasks:

1. Create Host Checker policies and set remediation options through the **Authentication > Endpoint Security > Host Checker** page of the Web console. For more details instructions, see:
 - “Securing managed vs. unmanaged endpoints using remediation” on page 5
 - “Securing endpoints using custom rules in Host Checker policies” on page 9
2. For Windows clients, determine whether you need to use a pre-authentication access tunnel between the clients and policy server(s) or resources. If necessary, create a `manifest.hcif` file with the tunnel definition and upload it through the **Authentication > Endpoint Security > Host Checker** page of the Web console. For instructions, see “Specifying Host Checker pre-authentication access tunnel definitions” on page 24.
3. Determine at which levels within the IVE framework you want to enforce the policies:
 - To enforce Host Checker policies when the user first accesses the IVE, implement the policies at the realm level by using the **Administrators > Admin Realms > Select Realm > Authentication Policy > Host Checker** or the **Users > User Realms > Select Realm > Authentication Policy > Host Checker** pages of the Web console.
 - To allow or deny users access to roles based on their compliance with Host Checker policies, implement the policies at the role level by using the **Administrators > Admin Roles > Select Role > General > Restrictions > Host Checker** or the **Users > User Roles > Select Role > General > Restrictions > Host Checker** pages of the Web console.
 - To map users to roles based on their compliance with Host Checker policies, use custom expressions in the **Administrators > Admin Realms > Select Realm > Role Mapping** or the **Users > User Realms > Select Realm > Role Mapping** pages of the Web console.
 - To allow or deny users access to individual resources based on their compliance with Host Checker policies, use conditions in the **Users > Resource Policies > Select Resource > Select Policy > Detailed Rules > Select|CreateRule** page of the Web console.
4. Specify how users can access the Host Checker client. Host Checker enforces the policies you define using a client-side agent.
 - To enable automatic installation of the Host Checker client-side agent on all platforms, use the **Administrators > Admin Realms > Select Realm > Authentication Policy > Host Checker** page or the **Users > User Realms > Select Realm > Authentication Policy > Host Checker** page of the Web console.

- To download the Host Checker installer and manually install it on your Windows users' systems, use the **Maintenance > System > Installers** page of the Web console.
5. Determine whether you want to create client-side logs. If you enable client-side logging through the **System > Log/Monitoring > Client-side Log** page of the Web console, the IVE appliance creates log files on your users' systems and writes to the file whenever Host Checker runs.

Securing managed vs. unmanaged endpoints using remediation

You can use Host Checker policies to distinguish between managed and unmanaged computers, and grant access accordingly. When defining Host Checker policies, you can also specify remediation actions that you want Host Checker to take if a user's computer does not meet the requirements of the policy.

This section describes how to grant access to the individual user roles and specify remediation actions based on the number of Host Checker policies that the user's computer passes, as described in the following table:

Table 1: Using Host Checker to set access for managed and unmanaged hosts

Type of Endpoint	Host Checker Status	Access Level
Managed	Passed all Host Checker policies	Provide full access
Managed	Passed managed host policies, but failed other Host Checker policies	Provide limited access or apply remediation actions
Unmanaged	Failed all managed host policies.	Force access through the Virtual Desktop

Topics in this section include:

- “Configuring Host Checker to secure managed endpoints” on page 6
- “Understanding Host Checker remediation” on page 7
- “Specifying remediation options” on page 7

Configuring Host Checker to secure managed endpoints

To configure Host Checker policies to distinguish between managed and unmanaged computers and grant access accordingly:

1. Create at least one Host Checker policy through the **Authentication > Endpoint Security > Host Checker** page of the Web console that determines whether an endpoint is a managed host. For instance, you can create a policy that checks whether the endpoint has the latest signature files for a certain antivirus program. For more information, see “Securing endpoints using custom rules in Host Checker policies” on page 9.

2. Specify remediation actions for the policies using settings in the **Remediation** section of the **Authentication > Endpoint Security > Host Checker > Select Policy > Host Checker Policy** page of the Web console. For more information, see “Understanding Host Checker remediation” on page 7.
3. Create a “full access role,” “partial access role,” and “unmanaged access role” through the **Users > User Roles** page of the Web console.
4. Configure the following requirements for the individual roles using settings in the **Users > User Roles > Select Role > General > Restrictions > Host Checker** page of the Web console:
 - **Full access role**—Select the **Allow users whose workstations meet the requirements specified by these Host Checker policies** option and then choose all Host Checker policies, including the managed host policy described in step 1.
 - **Partial access role**—Select the **Allow users whose workstations meet the requirements specified by these Host Checker policies** option and then choose the managed host policy described in step 1. You can also enforce a subset of the other Host Checker policies that evaluate the endpoint for partial access in case the endpoint does not pass the Host Checker policies for full access.
 - **Unmanaged access role**—Select the **Allow users whose workstations meet the requirements specified by these Host Checker policies** option and then choose the managed host policy described in step 1. If the endpoint does not pass this policy, you can display a remediation page and instruct the user to run a Virtual Desktop and then sign in again.
5. Create a user authentication realm through the **Users > User Realms** page of the Web console.
6. Navigate to the **Users > User Realms > Select Realm > Role Mapping** page of the Web console and select the **User must select from among assigned roles** option. Also use options in this page to create role mapping rules that specify the roles you want to make available to users.

Understanding Host Checker remediation

When defining a Host Checker policy, you can specify remediation actions that you want Host Checker to take if a user’s computer does not meet the requirements of the policy. For example, you can configure the IVE to display a remediation page to the user that contains your specific instructions and links to resources to help the user bring his computer into compliance with the Host Checker policy requirements.

For example, the user may see a remediation page that contains instructions and a link such as:

Your computer's security is unsatisfactory.

Your computer does not meet the following security requirements. Please follow the instructions below to fix these problems. When you are done click **Try Again**. If you choose to **Continue** without fixing these problems, you may not have access to all of your intranet servers.

1. TrendMicro

Instructions: You do not have the latest signature files. **Click here to download the latest signature files.**

2. ManagedPC

Instructions: Please sign in again from the Virtual Desktop.

For each Host Checker policy, you can configure two types of remediation actions:

- **User-driven**—Using custom instructions, you can inform the user about the failed policy and how to make his computer conform. The user must take action to successfully re-evaluate the failed policy. For instance, you can create a custom page that enables the user to link a policy server or Web page where the user can bring his computer into compliance.
- **Automatic (system-driven)**—You can configure Host Checker to automatically remediate the user's computer. For example, you can kill processes, delete files, or an alternate policy when the initial policy fails. On Windows, you can also call the `HCIF_Module.Remediate ()` API function as part of a J.E.D.I. DLL. Host Checker does not inform users when performing automatic actions. (You could, however, include information in your custom instructions about the automatic actions.)

Specifying remediation options

To specify remediation actions for a Host Checker policy:

1. Create a new Host Checker policy or select an existing one through the **Authentication > Endpoint Security > Host Checker** page of the Web console
2. Specify the remediation actions that you want Host Checker to perform if a user's computer does not meet the requirements of the current policy:
 - **Enable Custom Instructions**—Enter the instructions you want to display to the user on the Host Checker remediation page. You can use the following HTML tags to format text and add links to resources such as policy servers or web sites: `<i>`, ``, `
`, ``, and `<a href>`. For example:

You do not have the latest signature files.

`Click here to download the latest signature files.`



NOTE: For Windows clients, if you include in the instructions a link to an IVE-protected policy server, define a pre-authentication access tunnel. For information, see “Specifying Host Checker pre-authentication access tunnel definitions” on page 24.

- **Evaluate other policies**—You can select one or more alternate policies that you want Host Checker to evaluate if the user’s computer does not meet the current policy requirements. For example, if a user attempts to access the IVE from an outside client computer such as a kiosk, you can use this option to evaluate an alternate policy that requires the user to access the IVE in a Sygate Virtual Desktop environment. Select the alternate policy in the **HC Policies** list and then click **Add**.



NOTE: If you configure an alternate policy to use its own alternate policy, Host Checker does not evaluate that “second-level” alternate policy for the current policy. In other words, Host Checker only evaluates one alternate policy per transaction.

- **Remediate**—(Third party DLLs only) You can select this option to perform remediation actions specified via the **Remediate** () API function in a third-party J.E.D.I. DLL. For more information, see the “Using the Host Check Server Integration Interface” on page 21.
- **Kill Processes**—On each line, enter the name of one or more processes you want to kill if the user’s computer does not meet the policy requirements. You can include an optional MD5 checksum for the process. (You cannot use wildcards in the process name.) For example:

```
keylogger.exe
MD5: 6A7DFAF12C3183B56C44E89B12DBEF56
```

- **Delete Files**—Enter the names of files you want to delete if the user’s computer does not meet the policy requirements. (You cannot use wildcards in the file name.) Enter one file name per line. For example:

```
c:\temp\bad-file.txt
/temp/bad-file.txt
```

3. Click **Save Changes**.

Securing endpoints using custom rules in Host Checker policies

This section discusses how to use custom rules to implement specific endpoint solutions that are not preconfigured through the IVE’s predefined rules and policies. For more information about preconfigured rules and policies, see the *Juniper Networks Secure Access Administration Guide*.

This section contains the following information about using Host Checker to perform custom host checks:

- “Specifying custom rules in Host Checker policies” on page 9
- “Custom rule: Screen saver check” on page 13
- “Custom rule: Microsoft hotfix installation check” on page 13
- “Custom rule: Windows update status verification” on page 14

- “Custom rule: Network bridge disabled” on page 14
- “Custom rule: IP forwarding disabled” on page 15
- “Custom rule: Internet connection sharing disabled” on page 15
- “Custom rule: Google Desktop Search check” on page 15

Specifying custom rules in Host Checker policies

You can use custom rules within a Host Checker policy to define requirements that your users’ computers must meet. Using custom rules, you can:

- Configure Host Checker to check for custom DLLs that perform customized client-side checks.
- Verify that certain ports are open or closed on the user’s computer.
- Confirm that certain processes are or are not running on the user’s computer.
- Check that certain files are or are not present on the client machine.
- Evaluate the age and content of required files through MD5 checksums.
- Confirm that registry keys are set on the client machine.

To create a client-side Host Checker policy that uses custom rules:

1. Create a new policy through the **Authentication > Endpoint Security > Host Checker** page of the Web console.
2. Click the tab that corresponds to the operating system for which you want to specify Host Checker options—**Windows**, **Mac**, or **Linux**. In the same policy, you can specify different Host Checker requirements on each operating system. For example, you can create one policy that checks for different files or processes on each operating system.
3. Under **Rule Settings**, choose the **Custom: Process** or **Custom: Registry Setting** option and click **Add**. The **Add Custom Rule** page for the rule type appears. Configure the custom rule using instructions in the following sections:
 - “Configuring custom process rules” on page 10
 - “Configuring registry setting rules” on page 11

Note that these instructions only describe how to create process and registry check custom rules, since these are the only types of checks used in the custom rules in the sections that follow. For information about **3rd Party NHC Checks**, see the “Using the Host Check Client Interface” on page 18. For information about port and file checks, see the *Juniper Networks Secure Access Administration Guide*.

- Optionally add additional rules to the policy and specify how Host Checker should evaluate multiple rules. For instructions, see “Evaluating multiple rules in a single Host Checker policy” on page 12.

Configuring custom process rules

Use the process rule type to control the software that a client may run during a session. This rule type ensures that certain processes are running or not running on the client machine before the user can access resources protected by the IVE.

In the **Processes** configuration page:

- Enter a name for the process rule.
- Enter the name of a process (executable file), such as: `good-app.exe`.

You can use a wildcard character to specify the process name. For example:

`good*.exe`

You can use the following wildcards to specify a file name in an **Custom File** rule or a process name in an **Custom Process** rule:

Table 2: Wildcard characters for specifying a file name or process name

Wildcard Character	Description	Example
*	Matches any character	*.txt
?	Matches exactly one character	app-?.exe

- Select **Required** to require that this process is running or **Deny** to require that this process is not running.
- Specify the MD5 checksum value of each executable file to which you want the policy to apply (optional). For example, an executable may have different MD5 checksum values on a desktop, laptop, or different operating systems. On a system with OpenSSL installed—many Macintosh and Linux systems have OpenSSL installed by default—you can determine the MD5 checksum by using this command:

```
openssl md5 <processFilePath>
```

- Click **Save Changes**.

Configuring registry setting rules

Use the registry setting rule type to control the corporate PC images, system configurations, and software settings that a client must have in order to access the IVE. This rule type ensures that certain registry keys are set on the client machine before the user can access the IVE. You may also use registry checks to evaluate the age of required files and allow or deny access accordingly.

In the **Registry Settings** configuration page:

- Enter a name for the registry setting rule.

2. Select a root key from the drop-down list.
3. Enter the path to the application folder for the registry subkey.
4. Enter the name of the key's value that you want to require (optional). This name appears in the **Name** column of the Registry Editor.
5. Select the key value's type (**String**, **Binary**, or **DWORD**) from the drop-down list (optional). This type appears in the **Type** column of the Registry Editor.
6. Specify the required registry key value (optional). This information appears in the **Data** column of the Registry Editor.

If the key value represents an application version, select **Minimum version** to allow the specified version or newer versions of the application. For example, you can use this option to specify version information for an antivirus application to make sure that the client antivirus software is current. The IVE uses lexical sorting to determine if the client contains the specified version or higher. For example:

```
3.3.3 is newer than 3.3
4.0 is newer than 3.3
4.0a is newer than 4.0b
4.1 is newer than 3.3.1
```

7. Click **Save Changes**.



NOTE: If you specify only the key and subkey, Host Checker simply verifies the existence of the subkey folder in the registry.

Evaluating multiple rules in a single Host Checker policy

If you choose to include multiple rules within a single client-side policy, you must specify how Host Checker should evaluate those rules.

To specify requirements for multiple rules within a Host Checker policy:

1. In the Web console, choose **Authentication > Endpoint Security > Host Checker**.
2. In the **Policies** section of the page, click on an existing policy that includes multiple rules.
3. In the **Require** section, select one of the following options:
 - **All of the above rules**—Select this option to specify that the user's computer must return a success value for all of the policy's rules in order to gain access.
 - **Any of the above rules**—Select this option to specify that the user's computer must return a success value for any of the policy's rules in order to gain access.

- **Custom**—Select this option to customize the rules that the user’s computer must meet in order to gain access. Then, create the custom rule using instructions in the following step.
- 4. (Custom expressions only) If you want to use alternative sets of rules in the policy, combine rules with Boolean operators (**AND**, **OR**) using the following guidelines:
 - Enter the name of the rules in the **Rules expression** text box.
 - Use the **AND** operator to require two rules or sets of rules to return a true value.
 - Use the **OR** operator to require either of two rules or sets to return a true value.
 - Use parenthesis to combine sets of rules.

For example, you can use the following expression to require a personal firewall to run, and require either of two possible antivirus products to run:

ZoneLabsFirewall AND (McAfeeAntivirus OR NortonAntivirus)

- 5. Click **Save Changes**.

Custom rule: Screen saver check

You can use the following registry setting check rules in a Host Checker policy to automatically activate a screen saver and lock a host with a password after a period of inactivity. If the user leaves his computer unattended, the screen saver activates and prevents another user from using it.

This screen saver policy checks the registry to see if the screen saver is password protected and sets it to display a blank screen after 900 seconds (15 minutes) of inactivity.

Table 3: Custom: Registry Setting

For this setting:	Enter this value:
Registry Root key:	HKEY_CURRENT_USER
Registry Subkey:	Control Panel\Desktop\ScreenSaveTimeOut
Type:	string
Value:	900

Table 4: Custom: Registry Setting

For this setting:	Enter this value:
Registry Root key:	HKEY_CURRENT_USER
Registry Subkey:	Control Panel\Desktop\ScreenSaveActive
Type:	string
Value:	1

For configuration instructions, see “Specifying custom rules in Host Checker policies” on page 9.

Custom rule: Microsoft hotfix installation check

You can use predefined Host Checker rules to check for Windows operating systems and service packs on your users’ systems, as described in the *Juniper Networks Secure Access Administration Guide*. If you want to further configure Host Checker to check for specific Microsoft hotfix installations, however, you can use custom rules.

You can use registry setting check custom rules to check a Windows client’s registry for installed KB patches at the required level. The following rules are examples of checking for two specific patches.

Table 5: Custom: Registry Setting

For this setting:	Enter this value:
Registry Root key:	HKEY_LOCAL_MACHINE
Registry Subkey:	SOFTWARE\Microsoft\Updates\Windows XP\SP2\KB810217\
Type:	string

Table 6: Custom: Registry Setting

For this setting:	Enter this value:
Registry Root key:	HKEY_LOCAL_MACHINE
Registry Subkey:	SOFTWARE\Microsoft\Updates\Windows XP\SP2\KB821557\
Type:	string

For configuration instructions, see “Specifying custom rules in Host Checker policies” on page 9.

Custom rule: Windows update status verification

You can use the following registry setting check rule to check the Active Update configuration setting in a Windows client’s registry to verify that the Windows update configuration is correct.

Table 7: Custom: Registry Setting

For this setting:	Enter this value:
Registry Root key:	HKEY_LOCAL_MACHINE
Registry Subkey:	SOFTWARE\Microsoft\Windows\CurrentVersion\WindowsUpdate\Auto Update\AUState
Type:	dword
Value:	2

For configuration instructions, see “Specifying custom rules in Host Checker policies” on page 9.

Custom rule: Network bridge disabled

You can use the following registry setting check rule to verify that network bridging is disabled in Windows XP to ensure that a client cannot bridge traffic to another network while connected to the IVE.

Table 8: Custom: Registry Setting

For this setting:	Enter this value:
Registry Root key:	HKEY_LOCAL_MACHINE
Registry Subkey:	SYSTEM\CurrentControlSet\Services\BridgeMP\DisableForwarding
Type:	dword
Value:	1

For configuration instructions, see “Specifying custom rules in Host Checker policies” on page 9.

Custom rule: IP forwarding disabled

You can use the following registry setting check rule to verify that IP forwarding is disabled in Windows XP to ensure that a host is not allowed to forward (route) IP traffic.

Table 9: Custom: Registry Setting

For this setting:	Enter this value:
Registry Root key:	HKEY_LOCAL_MACHINE
Registry Subkey:	SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\IPEnableRouter
Type:	dword
Value:	0

For configuration instructions, see “Specifying custom rules in Host Checker policies” on page 9.

Custom rule: Internet connection sharing disabled

You can use the following registry setting check rule to verify that Internet connection sharing is disabled in Windows XP to ensure that a host is unable to share an Internet Connection, including a Network Connect connection.

Table 10: Custom: Registry Setting

For this setting:	Enter this value:
Registry Root key:	HKEY_LOCAL_MACHINE
Registry Subkey:	SYSTEM\CurrentControlSet\Services\SharedAccess\Start
Type:	dword
Value:	3

For configuration instructions, see “Specifying custom rules in Host Checker policies” on page 9.

Custom rule: Google Desktop Search check

Google Desktop Search creates a security concern for companies that provide remote access to their network. This concern is particularly important for SSL VPN sites since they readily provide remote access from most endpoints in a client-less fashion.

Google Desktop Search is an application composed of several processes that index content that a user views or saves. These processes enable a user to search for content from a web page hosted on the local computer. Google Desktop Search can search the full text of email, files, viewed web pages, and chats. Specifically, users can:

- Search email from Outlook 2000 + and Outlook Express 5 +
- Search files in TXT, HTML, DOC, XLS, and PPT formats (Office 2000 +)
- Search chats from AOL 7 + and AOL Instant Messenger 5 +
- Search web pages viewed in Internet Explorer 5 +

You can create a Host Checker policy that checks for the Google Desktop Search processes and displays a HTML remediation page. This HTML page can either prompt the user to launch a virtual desktop and/or inform the user that you are killing the Google Desktop Search processes.

You can use the following process check rules to check whether Google Desktop Search is currently running on the workstation. If it is, you can deny access and configure remediation actions.

Table 11: Custom: Process

For this setting:	Enter this value:
Process Name:	googledesktop.exe
Access Setting:	Deny

Table 12: Custom: Process

For this setting:	Enter this value:
Process Name:	googledesktopindex.exe
Access Setting:	Deny

Table 13: Custom: Process

For this setting:	Enter this value:
Process Name:	googledesktopcrawl.exe
Access Setting:	Deny

To configure a Host Checker policy with remediation actions for the Google Desktop Search process check rules, use instructions in “Specifying remediation options” on page 7. When configuring the remediation options, select **Kill Process** and then enter the names of the three Google Desktop Search processes on each line. Note that the user must have the required privileges to kill the processes.

Securing endpoints using the Sygate Security Agent

This section describes the recommended configuration of the IVE and the Sygate Security Agent for interoperability and optimal security.

Configuring the Sygate Management Server

The only required configuration you must perform on the Sygate Management Server is to create Host Integrity rule(s) and enable the Host Integrity checks for specific groups.

Configuring the Sygate Security Agent

You must configure the Sygate Security Agent to look up the Sygate Management Server(s) using a fully-qualified DNS domain name only. If you use a private IP address (such as 10.0.0.1) in the `sylink.xml` file, then the Sygate Security Agent cannot connect to the Sygate Management Server(s) via the IVE’s SSL VPN or Secure Application Manager (SAM).

Clients connected to the IVE use the policy defined in the Remote location.

When configuring the `sylink.xml` file, set the `ConfigControl FreezeSMSList` value to 1. Otherwise, the Sygate Management Server overwrites the null IP address in the defined Sygate Management Server IP/name pairs. In subsequent communications, the client uses the IP and not the name to contact the server, thereby breaking the connection.

Figure 1: Sample client `sylink.xml` file

```
<?xml version="1.0" encoding="UTF-8" ?>
<ServerSettings>
  <ProfileControl ByServer="1" />
  <ConfigControl FreezeSmsList="1" FreezeSmsListOrder="0" />
  <RefreshSeconds>600</RefreshSeconds>
  <TimeoutSeconds>600</TimeoutSeconds>
  <RegisterClient BasedOnUserLogon="0" />
  <Server Ip="0" Name="smsserver1.corp.infopeople.ca" HttpPort="80" HttpsPort="443" />
  <Server Ip="0" Name="smsserver2.corp.infopeople.ca" HttpPort="80" HttpsPort="443" />
</ServerSettings>
```

As an alternative, you can also use an Internet-accessible IP address of the Sygate Management Server.



NOTE: You must configure the heartbeat interval in the `sylink.xml` file since the Sygate Management Server blocks it from updates.

Chapter 3

Using J.E.D.I. interfaces to implement endpoint solutions

If you want to implement your own third-party endpoint solution and cannot use the methods described in the previous chapter, you can use the J.E.D.I. APIs provided with the product.



NOTE: The information in this chapter applies to Host Checker running on Windows clients only.

An IVE appliance provides two different APIs that you can use to integrate third-party functionality:

- **Host Check Client Interface (Windows only)**—The Host Check Client Interface is an API that allows you to run your own DLLs using Host Checker. Through the interface, you can prompt Host Checker to run a DLL that you already installed on the user’s system or distributed as part of a corporate OS image, including programs that check compliance with corporate images, antivirus software, and personal firewall clients. Host Checker runs the specified DLL when a user signs into the IVE, and then bases its subsequent actions on the success or failure result that the DLL returns. For example, you may deny a user access to the IVE if the client check software fails. For more information, see “Using the Host Check Client Interface” on page 18.
- **Host Check Server Integration Interface (Windows only)**—The Host Check Server Integration Interface is an API that allows you to tightly integrate a J.E.D.I. compliant system with the IVE. Like the Host Check Client Interface, you can use the Host Check Server Integration Interface to prompt Host Checker to run third-party software on the client, including host integrity scans, malware detectors, and virtual environments. With this interface, you may also specify with granularity what Host Checker should do based on the result of the diverse policy checks that the third-party applications conduct. You can invoke these policies to dynamically map users to realms, roles, and resources based on the results of individual policies contained in your software package. For information, see “Using the Host Check Server Integration Interface” on page 21.

For more information on the Host Check interfaces, see:

- “Choosing which interface to use” on page 18
- “Using the Host Check Client Interface” on page 18
- “Using the Host Check Server Integration Interface” on page 21

Choosing which interface to use

The following table outlines when to use the Host Check Client Interface vs. the Host Check Server Integration Interface.

Table 14: Host Check client interface vs. server integration interface

Use the client interface to:	Use the server interface to:
<ul style="list-style-type: none"> ■ Enforce simple Host Checker policies. For example, you can use the client interface to return a yes/no value for a question such as “Is the proper software installed, running, or compliant?” 	<ul style="list-style-type: none"> ■ Enforce multiple Host Checker policies and/or hierarchical Host Checker policies.
<ul style="list-style-type: none"> ■ Reference DLLs that are already installed on the user’s system 	<ul style="list-style-type: none"> ■ Deploy third-party applications to the user’s system ■ Integrate custom DLLs or software onto the IVE appliance
<ul style="list-style-type: none"> ■ Enforce policies that do not require logging 	<ul style="list-style-type: none"> ■ Log results from your policy checks

Using the Host Check Client Interface

The Host Check Client Interface involves communicating with a third-party endpoint security application through its API and examining the return values to verify the trustworthiness of the client machine. The IVE platform provides a generic API library in the C programming language to support endpoint security applications that do not have an API. This Windows API is called the Host Check Client Interface API and contains the `NHC_EndpointSecure()` function, which checks the endpoint configuration.

Host Check Client Interface integration typically includes these steps:

1. An IVE administrator enables Host Checker for the desired realm, role, or resource. For this realm, role, or resource, the administrator specifies a 3rd party NHC check rule on the Web Console's **Host Checker** page. This rule specifies the location of your custom DLL on a client machine.
2. The IVE appliance downloads an ActiveX installer with the Host Check Client Interface package to the client machine of an authenticated user who is trying to access the realm, role, or resource.

3. The Host Check Client Interface package loads your custom DLL from the DLL location on the client. Before calling the `NHC_EndpointSecure()` function, the package calls the Windows `WinVerifyTrust()` function to validate the DLL's digital signature. (See "Signing your custom DLL" on page 19 for information about the user experience.)
4. The Host Check Client Interface package calls the `NHC_EndpointSecure()` function. If the function returns `NHC_STATUS_SECURE`, the third-party product endpoint security check succeeds and the IVE appliance maps the user to the realm, role, or resource. If the endpoint security check fails for a realm-level policy, the user sees an error stating that the computer does not comply with the endpoint security policy, and the user is redirected to the sign-in page. If you specify the URL to a failure page, this page opens in another browser window. If the endpoint security check fails for a role or resource level policy, the IVE appliance simply does not map the user to the role or resource.

This section contains the following information about the Host Check Client Interface:

- "Signing your custom DLL" on page 19
- "Deploying and maintaining your custom DLL" on page 20
- "NetScreen Host Checker (NHC) API" on page 20

Signing your custom DLL

You must digitally sign your custom DLL to ensure content integrity. Prior to calling your DLL, Host Checker calls the Windows `WinVerifyTrust()` function to attempt to verify the trustworthiness of the DLL.

- If your DLL is not digitally signed and the user's browser security settings are Medium-to-High, a message informs the user that the DLL is not signed and the Windows `WinVerifyTrust()` function cannot verify it as trustworthy. The user may choose to proceed, in which case Host Checker calls the DLL. If the `NHC_EndpointSecure()` function returns `NHC_STATUS_SECURE`, the IVE maps the user to the realm, role, or resource. If the user chooses to cancel the operation, the IVE does not map the user to the realm, role, or resource.
- If the DLL is digitally signed but `WinVerifyTrust()` cannot verify the trustworthiness of the DLL, a message informs the user. The user may choose to proceed, in which case the Host Checker calls the `NHC_EndpointSecure()` function. If this function returns `NHC_STATUS_SECURE`, the IVE maps the user to the realm, role, or resource. If the user chooses to cancel the operation, the IVE appliance does not map the user to the realm, role, or resource.
- If the DLL is digitally signed and `WinVerifyTrust()` verifies the trustworthiness of the DLL, a message informs the user that the content provider and its integrity have been verified and asked if he wishes to proceed. If the user proceeds, Host Checker calls `NHC_EndpointSecure()`. If this function returns `NHC_STATUS_SECURE`, the IVE maps the user to the realm, role, or resource.

Deploying and maintaining your custom DLL

To deploy your custom DLL, you need to:

- At the system level, configure the Host Checker feature to call your custom DLL and specify the path (including the file name) to where the DLL is stored on client machines.
- Install the DLL on the appropriate client machines or distribute the DLL as part of the corporate PC image.
- Create a low-privilege realm or role that users can access if the endpoint security check fails and make the DLL available on a page configured as an IVE bookmark for those users.
- Check the DLL timestamp to ensure that the version is current. If the endpoint security check fails, redirect users to a "safety page" that enables them to authenticate and download the current DLL.

NetScreen Host Checker (NHC) API

This section contains function definitions for the NHC API.

NHC API Definitions

```
#define NHC_STATUS_SECURE      1
#define NHC_STATUS_FAILURE    -1
#define NHC_STATUS_UNSECURE   -2
#define NHC_STATUS_BADPARAMETER -3
```

NHC_EndpointSecure() Required

The NHC_EndpointSecure function verifies the security of the endpoint based on the criteria established by the implementer.

Syntax

```
NHC_API int WINAPI NHC_EndpointSecure(void);
```

Parameters

None

Return Values

- NHC_STATUS_SECURE—The endpoint client is secure
- NHC_STATUS_UNSECURE—The endpoint client is not secure
- NHC_STATUS_FAILURE—The endpoint application is not running
- NHC_STATUS_BADPARAMETER—An internal error has occurred; the endpoint client should be judged insecure

C Header file: NHC.h

Include this header file in your DLL:

```
#ifndef NHC_EXPORTS
#define NHC_API __declspec(dllexport)
#else
#define NHC_API __declspec(dllimport)
#endif

#define NHC_STATUS_SECURE      1
#define NHC_STATUS_FAILURE    -1
#define NHC_STATUS_UNSECURE   -2
#define NHC_STATUS_BADPARAMETER -3

#ifdef __cplusplus
extern "C" {
#endif
NHC_API int NHC_EndpointSecure(void);
#ifdef __cplusplus
}
#endif
```

Using the Host Check Server Integration Interface

The Host Check Server Integration Interface contains a generic API library in the C++ programming language. This section describes the API and how to implement it for use with Host Checker.

This section contains the following information about the Host Check Server Integration Interface:

- “Deploying third party applications through Host Checker” on page 22
- “Creating your interface DLL” on page 27

Deploying third party applications through Host Checker

To deploy third party applications through Host Checker, you must complete the steps defined in the following sections:

1. “Creating an endpoint security package” on page 22
2. “Uploading the package to the IVE” on page 26
3. “Configuring Host Checker to use the package” on page 26

Creating an endpoint security package

An endpoint security package is a collection of files that comprise your third party functionality. When creating a package, you must include the following types of files:

- **Interface DLL**—An interface DLL is a program that carries out your specialized functions on the client. When creating your DLL, you must provide an interface between your modules and Host Checker using functions defined in the Host Check Server Integration Interface.
- **Package definition file**—A package definition file, which must be named `MANIFEST.HCIF`, defines the name of your interface DLL, the Host Checker policies defined in your DLL, and any optional pre-authentication access tunnel definitions. For more information, see “Creating the `MANIFEST.HCIF` package definition file” on page 23. Note that if you do not include policies in your package, Host Checker simply enforces that the package has run on the client. If you do declare policies through this file, they become available through the IVE Web console where you can implement them at the realm, role, and resource policy levels.
- **Host Check Server Integration Interface header file**—The Host Check Server Integration Interface header file (`hcif.h`) defines the Host Check Server Integration Interface functions that you must use in your DLL. You must add this file to your package’s `include` folder in order to use the functions that are provided with the Host Check Server Integration Interface. You can find a copy of this file in the SDK provided with the product.

In addition to these required files, you may also include your own data files in the package.

This section contains the following information about creating an endpoint security package:

- “Creating the `MANIFEST.HCIF` package definition file” on page 23
- “Using access tunnels to make policy servers available to Host Checker clients” on page 23
- “Specifying Host Checker pre-authentication access tunnel definitions” on page 24

Creating the `MANIFEST.HCIF` package definition file

Within the `MANIFEST.HCIF` package definition file, you must include one definition per line, with a blank line between each definition, using the following format:

```
HCIF-Main : <DLLName>
```

```
HCIF-Policy : <PolicyName>
```

```
HCIF-IVE-Tunnel: <client-loopback>:port <policy-server>:port
```

where:

<DLLName> is the name of the interface DLL, such as `myPestPatrol.dll`.

<PolicyName> is the name of a policy defined in the DLL, such as myFileCheck. You can define multiple policies by using the HCIF-Policy statement for each policy.

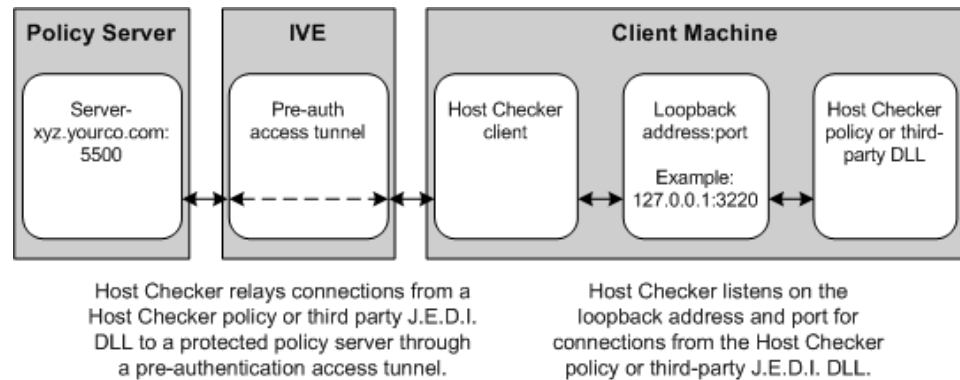
HCIF-IVE-Tunnel: <client-loopback>:port <policy-server>:port is an optional pre-authentication access tunnel definition. For information on the syntax of a tunnel definition, see “Using access tunnels to make policy servers available to Host Checker clients” on page 23.

Using access tunnels to make policy servers available to Host Checker clients

If your policies require Host Checker methods or third-party J.E.D.I. DLLs to access a policy server (or other resource) to check compliance before users are authenticated, you can use one of the following methods to make the resource available to the Host Checker Windows clients:

- **Deploy the policy server in a DMZ where Host Checker methods or third-party J.E.D.I. DLLs can access the server directly instead of going through the IVE**—This is the simplest deployment because you do not have to define a Host Checker pre-authentication access tunnel through the IVE between clients and the policy server.
- **Deploy the policy server in a protected zone behind the IVE (Windows only)**—This deployment requires you to define a *pre-authentication access tunnel*. A pre-authentication access tunnel enables Host Checker methods or third-party J.E.D.I. DLLs to access the IVE-protected policy server or resource before the IVE authenticates users. To define a pre-authentication access tunnel, you associate a loopback address (or host name) and port on the client with an IP address and port on the policy server. You add one or more tunnel definitions to the MANIFEST.HCIF file, which you then upload to the IVE. You can upload multiple MANIFEST.HCIF files to the IVE. For all third-party policies enabled on a realm, Host Checker creates tunnels for all of the tunnel definitions in all of the MANIFEST.HCIF files, assuming the definitions are unique. For configuration instructions, see “Specifying Host Checker pre-authentication access tunnel definitions” on page 24.

While running on a Windows client, Host Checker listens for a connection on each loopback address and port you specify in the tunnel definitions. The connections can originate from the integrated Host Checker methods and from client-side or server-side J.E.D.I. DLLs. Host Checker uses the pre-authentication access tunnel(s) to forward the connections through the IVE to the policy server(s) or other resource.

Figure 2: Host Checker creates a tunnel from a client to a policy server behind the IVE

NOTE: Host Checker pre-authentication access tunnels are supported on Windows only.

Specifying Host Checker pre-authentication access tunnel definitions

A definition for a Host Checker pre-authentication access tunnel configures access to one policy server or other resource. Each tunnel definition consists of a pair of IP addresses and ports: one loopback IP address and port on the client, and one IP address and port on the policy server. You specify one or more tunnel definition(s) in the MANIFEST.HCIF package definition file.

The syntax of a pre-authentication access tunnel definition is:

```
HCIF-IVE-Tunnel: <client-loopback>:port <policy-server>:port
```

where:

<client-loopback> is a loopback address that begins with 127. and takes any of the following forms:

- An IP address and port that takes the form of 127.*.*.*:port. To avoid conflicts with JSAM, do not use 127.0.0.1 with port 80, but you can use 127.0.0.1 with other ports. For example: 127.0.0.1:3220
- A host name that resolves to a loopback address that begins with 127. You can use a local hosts file on each client computer or a DNS server to resolve the loopback address.
- A host name that does not resolve to a loopback address, or it resolves to a non-loopback address. In these cases, Host Checker allocates a loopback address and updates the local hosts file on the client with the mapping. Note that the user must have administrator privileges in order for Host Checker to modify the local hosts file. If the user does not have administrator privileges, Host Checker cannot update the hosts file and cannot open the pre-authentication access tunnel. In that case, Host Checker logs an error.

<policy-server> is the IP address or host name of the back-end policy server. The IVE resolves the host name you specify.

For example, in the following tunnel definition, **127.0.0.1:3220** is the client loopback address and port, and **mysygate.company.com:5500** is the policy server host name and port:

```
HCIF-IVE-Tunnel: 127.0.0.1:3220 mysygate.company.com:5500
```

Or you can use a host name for the client, as in this example:

```
HCIF-IVE-Tunnel: mysygate.company.com:3220 mysygate.company.com:5500
```

Keep the following in mind when specifying tunnel definitions:

- You must add a blank line between each line in the **MANIFEST.HCIF** file, and you can use a semi-colon at the beginning of a line to indicate a comment. For example:

```
HCIF-Main : myPestPatrol.dll
```

```
HCIF-Policy : myFileCheck
```

```
HCIF-Policy : myPortCheck
```

```
; Tunnel definitions
```

```
HCIF-IVE-Tunnel: 127.0.0.1:3220 mysygate.company.com:5500
```

```
HCIF-IVE-Tunnel: 127.1.1.1:3220 mysygate2.company.com:5500
```

```
HCIF-IVE-Tunnel: mysygate.company.com:3220 mysygate3.company.com:5500
```

- If <client-loopback> is a non-loopback address, then Host Checker cannot open the pre-authentication access tunnel and logs an error instead.
- If you use a loopback address other than **127.0.0.1** (such as **127.0.0.2** and above), clients who are using Windows XP Service Pack 2 must install the XP SP2 Hot Fix. See:

```
http://support.microsoft.com/default.aspx?scid=kb;en-us;884020
```

- Your DLL must use the same <client-loopback> address and port or host name that you specify in the **MANIFEST.HCIF** file.
- Because a pre-authentication access tunnel is open only while Host Checker is running, your DLL can access its IVE-protected policy server only while Host Checker is running.
- If your DLL uses **HTTPS** to connect to its policy server via a host name that resolves properly to the loopback address, no server certificate warnings appear. However, if the your DLL connects explicitly via a loopback address, then server certificate warnings do appear because the host name in the certificate does not match the loopback address. (Your DLL can configure the DLL to ignore these warnings.)

Uploading the package to the IVE

In order for the IVE to recognize a package definition file, you must name it `MANIFEST.HCIF` and include it a folder named `META-INF`. You must then archive the `META-INF` folder containing the `MANIFEST.HCIF` file along with the interface DLL and any initialization files in one zip file for the policy package. For example, the contents of the zip file for a policy package can be:

```
META-INF/MANIFEST.HCIF
hcif-myPestPatrol.dll
hcif-myPestPatrol.ini
```

You can upload multiple policy packages each containing a different `MANIFEST.HCIF` file to the IVE.

Host Checker creates tunnels for all of the tunnel definitions in all of the `MANIFEST.HCIF` files, assuming the definitions are unique.

Then you upload the package zip file to the IVE through the **Authentication > Endpoint Security > Host Checker** page of the Web console. Note that after you upload a Host Checker policy package to the IVE, you cannot modify the package contents on the server. Instead, you must modify the package on your local system and then upload the modified version to the IVE.

Configuring Host Checker to use the package

After you upload a valid package, the IVE automatically exposes the policies contained in the package in the realm, role, and resource policy configuration pages of the Web console. You may then use these pages to implement the policies. When a user tries to access a realm, role, or resource protected by one of these policies, the IVE runs the program that you have uploaded to the server.

Creating your interface DLL



NOTE: Version 5 of the IVE supports a new version 2 of the Host Checker API that contains the following changes:

- The `HCIF_Module.checkPolicy()` function is a new function that replaces the `HCIF_Check()` function in version 1.
- The `HCIF_Module.remediate()` function is new.

Version 5 of the IVE also supports version 1 of the Host Checker API. For information on specifying the version you want to use, see “`HCIF_Module.version()` function” on page 29.

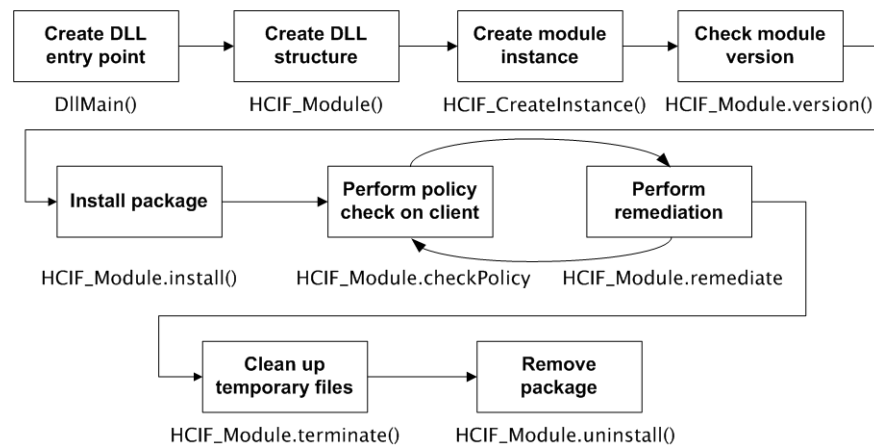
When you create an interface DLL to carry out your specialized functions on user machines, you must use the functions provided in the Host Check Server Integration Interface to:

1. Provide an entry point for the DLL, as explained in “`DllMain()` function” on page 28.

2. Create a module structure for your DLL, as explained in “HCIF_Module() function” on page 28
3. Create an instance for your module, as explained in “HCIF_CreateInstance() function” on page 29.
4. Specify the API version you want to use with your module, as explained in “HCIF_Module.version() function” on page 29.
5. Install your endpoint security package on the user’s machine, as explained in “HCIF_Module.install() function” on page 29.
6. Perform endpoint security checks on individual policies, as explained in “HCIF_Module.checkPolicy() function (version 2 only)” on page 29. This function loops with the HCIF_Module.remediate() function.
7. (Optional) Perform remediation, as explained in “HCIF_Module.remediate() function (version 2 only)” on page 30. This function loops with the HCIF_Module.checkPolicy() function.
8. Stop the DLL and clean up temporary files, as explained in “HCIF_Module.terminate() function” on page 30.
9. Remove your package from the user’s machine, as explained in “HCIF_Module.uninstall() function” on page 30.

The following diagram illustrates the steps and corresponding functions:

Figure 3: Creating your interface DLL (Host Checker API version 2)



NOTE:

- The hcif.cpp sample file in the SDK includes all of the functions described here.
- You can test whether your package properly conforms to the standards described here using the hciftool.exe tool provided with the SDK.

DllMain() function

Use the `DllMain()` function to define an entry point for your DLL within the main Host Checker framework. This function alerts Host Checker that it needs to execute your third party DLL. When adding the `DllMain()` function to your project, copy and use the exact version that is in the `hcif.cpp` sample that is included with the SDK.

HCIF_Module() function

Use the `HCIF_Module()` function to create a structure for your DLL. You should call all of the other functions described in this section from within the `HCIF_Module()` function.

HCIF_CreateInstance() function

Use the `HCIF_CreateInstance()` function to create an individual instance of your service and to create a DLL entry point. The Host Checker framework calls the `HCIF_CreateInstance()` function to obtain a pointer to the `HCIF_Module` structure.

HCIF_Module.version() function

Use the `HCIF_Module.version()` function to return the Host Checker API version that was used to compile the module. The Host Checker framework uses the return value to determine whether to call the `HCIF_Module.checkPolicy()` function or the `HCIF_Check()` function. If you specify the version return value is 2 (which is the default value defined in the `HCIF_API_Version` constant), the Host Checker framework calls the `HCIF_Module.checkPolicy()` function. If you specify the version return value is 1, the Host Checker framework calls the `HCIF_Check()` function. For backwards compatibility, the framework only calls the `HCIF_Check()` function for a version 1 interface DLL.

HCIF_Module.install() function

Use the `HCIF_Module.install()` function to deliver required files from the IVE to the user's system. Within the `HCIF_Module.install()` function, you may use the standard C `getFile` function to retrieve files that you have uploaded to the IVE and copy them to Host Checker's default directory on the user's system:

```
C:\Documents and Settings\\Application Data\Juniper Networks\Host Checker
```

HCIF_Check() function (version 1 only)

For an interface DLL that uses version 1 of the Host Checker API, use the `HCIF_Check()` function to perform your client-side checks. You may use the `HCIF_Check()` function to run the core execution modules in your package. These modules should check the compliance of the endpoint with the configured policies and return a value of `TRUE` if the check was successful or `FALSE` if the check failed.

HCIF_Module.checkPolicy() function (version 2 only)

For an interface DLL that uses version 2 of the Host Checker API, use the `HCIF_Module.checkPolicy()` function to perform client-side checks using individual policies. For example, suppose there are two policies named `Policy-1` and `Policy-2`. The framework calls this function for `Policy-1`, and then it calls this function again for `Policy-2` individually. That is, you can call an individual policy and run a specific set of code for that policy only.

This function should check the compliance of the endpoint with the configured individual policies and call `HCIF_Service report()` to report the status of each individual policies. The framework calls this function periodically during a session. This function returns a value of `TRUE` if the check was successful or `FALSE` if the check failed.

**NOTE:**

- Only those policies enabled on a realm are passed to the `HCIF_Module.checkPolicy()` function.
 - The framework uses the return value of the `HCIF_Module.checkPolicy()` function as the status of the overall Host Checker policy that contains the individual J.E.D.I. third-party policies.
-

HCIF_Module.remediate() function (version 2 only)

For an interface DLL that uses version 2 of the Host Checker API, Host Checker passes a list of failed policies to the `HCIF_Module.remediate()` function. Use the list of failed policies in the `HCIF_Module.remediate()` function to perform automatic or user-driven remediation actions to fix the policy failures. The framework only calls this function if the **Remediate** option is enabled on the associated Host Checker policy. (This option is on the **Authentication > Endpoint Security > Host Checker** page of the Web console). This function returns a value of `TRUE` if the remediation was successful, or `FALSE` if the remediation failed.



NOTE: The return value of the `HCIF_Module.remediate()` function does *not* affect the status of the overall Host Checker policy that contains the individual J.E.D.I. third-party policies.

HCIF_Module.terminate() function

Use the `HCIF_Module.terminate()` function to perform any final clean up at the end of a session, including freeing resources, removing temporary files, or reverting registry entries. The Host Checker framework calls the `HCIF_Module.terminate()` function at the end of the user's IVE session.

HCIF_Module.uninstall() function

Use the `HCIF_Module.uninstall()` function to remove all traces of your module from the user's system. The Host Checker framework calls the `HCIF_Module.uninstall()` function after the `HCIF_Module.terminate()` function.

