## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>7</td>
</tr>
<tr>
<td>Principles Behind Best Practices</td>
<td>7</td>
</tr>
<tr>
<td>Third Party Testing</td>
<td>8</td>
</tr>
<tr>
<td>Related Security Documentation</td>
<td>8</td>
</tr>
<tr>
<td>Security Background Information</td>
<td>9</td>
</tr>
<tr>
<td>Disclaimer</td>
<td>9</td>
</tr>
<tr>
<td>Before You Begin</td>
<td>11</td>
</tr>
<tr>
<td>Determining Your Security Posture</td>
<td>11</td>
</tr>
<tr>
<td>What You Need Before You Start</td>
<td>13</td>
</tr>
<tr>
<td>Overview of Best Practices</td>
<td>14</td>
</tr>
<tr>
<td>Using Basic Infrastructure Protections</td>
<td>19</td>
</tr>
<tr>
<td>B2B Network Architecture</td>
<td>19</td>
</tr>
<tr>
<td>Additional Considerations for Reverse Invoke</td>
<td>23</td>
</tr>
<tr>
<td>Specifying Allowed Services from the Reverse Invoke Proxy Port</td>
<td>23</td>
</tr>
<tr>
<td>Preventing External Access to DSPs on the Internal Server</td>
<td>25</td>
</tr>
<tr>
<td>Selecting a Secure Transmission Protocol</td>
<td>26</td>
</tr>
<tr>
<td>Running Your webMethods Integration Server on a UNIX–Based System</td>
<td>27</td>
</tr>
<tr>
<td>Running Your webMethods Integration Server on a Windows System</td>
<td>30</td>
</tr>
<tr>
<td>Port Selection for Brokers</td>
<td>30</td>
</tr>
<tr>
<td>Choosing the Best Ports</td>
<td>30</td>
</tr>
<tr>
<td>Integration Server Authentication Mechanisms</td>
<td>31</td>
</tr>
<tr>
<td>User Name/Password Authentication</td>
<td>31</td>
</tr>
<tr>
<td>Private Key/Certificate Authentication</td>
<td>33</td>
</tr>
<tr>
<td>Integration Server Access Control Mechanisms</td>
<td>34</td>
</tr>
<tr>
<td>Controlling by IP Address or Domain</td>
<td>37</td>
</tr>
<tr>
<td>Understanding Users, Groups, ACLs, Folders, Services, and DSPs</td>
<td>39</td>
</tr>
<tr>
<td>Controls for Services</td>
<td>39</td>
</tr>
<tr>
<td>Controls for DSPs and Public Files</td>
<td>41</td>
</tr>
<tr>
<td>Controls for Web Applications</td>
<td>43</td>
</tr>
<tr>
<td>Controls for Aliases</td>
<td>43</td>
</tr>
<tr>
<td>Setting Correct ACLs on Services</td>
<td>43</td>
</tr>
<tr>
<td>ACL Settings for Custom Services</td>
<td>44</td>
</tr>
</tbody>
</table>
webMethods Fabric 6.5 Security Best Practices

Access to Resources ................................................................. 74
Further Reading ................................................................. 75

Auditing and Forensics ......................................................... 76
Digital Signatures ................................................................. 78
FIPS Support ................................................................. 80
Contacting webMethods ......................................................... 81

Appendix A. Externally Visible Services ......................................... 82
Appendix B. Special Considerations for Trading Networks ......................... 84
  Additional Groups and ACLs .................................................. 84
  Web Manager .......................................................... 84
  List, Read, Write, and Execute ACLs ...................................... 85
  Profile Management .......................................................... 85
  Document Security .......................................................... 86
  Security Certificates .......................................................... 87
  Polling .......................................................... 88
    When Your Server Polls a Partner’s Server ............................. 88
    When Your Server Is Being Polled ..................................... 89
  Reliable Tasks .......................................................... 90

Appendix C. Common Criteria .................................................. 92
  Scope of the Evaluation ..................................................... 92
  Assumptions for the Evaluated Configuration ............................. 94
  Installation .......................................................... 95
  Integration Server .......................................................... 96
  Broker .......................................................... 98
  Developer .......................................................... 99
  JDBC Adapter and JMS Adapter .............................................. 100
  Supplementary Information for Logging Guide ............................ 102
  Network Protocols and Architectures in the Evaluated Configuration .... 102
  Auditing in the Evaluated Configuration ................................ 104
  Operating System and JVM Requirements ................................ 105
  Configuring the Integration Server ....................................... 106
    Required Configuration Settings .................................... 106
    Users, Groups, and Passwords ........................................ 106
    ACLs .......................................................... 107
  Custom Flow and Java Services ........................................... 107
Broker Configuration ................................................................. 107
Configuring the Broker .............................................................. 108
Introduction

Security is a critical part of your Integration Platform. This document provides a set of “best practices” to ensure that your Integration Platform is configured in the most secure way possible for your environment.

There is no single best configuration for all environments. Your optimal configuration will depend on your business needs, the trade-offs between security and functionality, and the effort you want to spend implementing security capabilities. This document does not contain everything you need to know about security. Rather, it serves as a supplement to other webMethods documents and your organization’s own security practices.

This document covers best practices for webMethods Integration Platform 6.5. Users of other webMethods releases should contact webMethods Technical Services for the best practices information for those releases.

Note: The instructions in this document will help you update the configuration of a single server. If you have multiple servers (as in a cluster configuration, or a test system and a production system), see “Propagating Changes Across webMethods Integration Servers” on page 70 of this document for information about propagating the changes from one server to another.

Note: The Integration Server is the primary focus for security in the webMethods Integration Platform; therefore, most of this document focuses on the Integration Server.

Principles Behind Best Practices

This document is based on a small set of widely accepted security principles:

- System access should be permitted only where explicitly granted; the default should be to deny access.
- Possible conflicts in permissions should be minimized.
- The system configuration should be as simple as possible.
- User groups should be set up for administrators, developers, and partners—to clearly separate system permissions—thus reducing the risk that an unauthorized user will gain access to a restricted capability.
- Users and software should be granted the fewest privileges necessary to accomplish their tasks.
Third Party Testing

webMethods has engaged @stake to perform a review of the webMethods 6.5 security architecture for the Integration Server and Broker, combined with a penetration test. The results of their study are available upon request from security@webMethods.com. One of @stake’s key recommendations is to use this document to safely configure and operate your webMethods software.

Related Security Documentation

A brief overview of the webMethods Integration Platform can be found in the webMethods 6 Security White Paper, available from your webMethods sales representative or on webMethods Advantage.

Along with this document, you should be familiar with the following chapters from the webMethods Integration Server Administrator’s Guide:

- Chapter 4: “Using the Integration Server Administrator.” This chapter covers security information every administrator needs to understand.
- Chapter 5: “Managing Users and Groups.” This chapter describes how to set up users and groups, and their relationship to Access Control Lists (ACLs).
- Chapter 6: “Configuring the Server.”
  - “Configuring Ports” includes descriptions of how to configure the security parameters for each type of port.
  - “Specifying a Third–Party Proxy Server for Outbound Requests” describes how to configure your webMethods Integration Server if you are using a proxy server, whether the proxy server is a webMethods Integration Server or some other type of proxy.
- Chapter 7: “Managing Server Security.” This key chapter describes how to use key security mechanisms such as ACLs, how to set up SSL, and how to limit access to services by port.
- Chapter 8: “Using an External Directory (LDAP or NIS).” This chapter describes how to configure authentication to use an LDAP or NIS server.
- Appendix A: “webMethods Integration Server Deployment Checklist.” This appendix includes a checklist for fielding a webMethods Integration Server. In particular, stage 7 includes useful security hints.
Appendix B: “Server Configuration Parameters.” The section entitled “watt.security” identifies configurable security parameters stored by the webMethods Integration Server.

Security Background Information

You may also find the following resources useful in understanding security concepts:


Disclaimer

This document is provided for informational purposes only, and is not intended to, nor shall it create any warranty, either express or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose.

This document does not constitute “documentation” for purposes of any Software License Agreements between webMethods and its customers and partners.

webMethods shall not be liable for any damages, including but not limited to indirect, direct, special, incidental, or consequential damages arising out of this document or the information contained herein, even if webMethods has been advised of the possibility of
such damages, and notwithstanding any failure of essential purpose of any limited remedy of any kind.

Mention of a product in this document does not imply endorsement by webMethods. webMethods Broker, webMethods Developer, webMethods Integration Server, webMethods Manager, and webMethods Trading Networks are trademarks of webMethods, Inc. webMethods and the webMethods logo are registered trademarks of webMethods, Inc. All other marks are the property of their respective owners.
Before You Begin

This section identifies several issues to consider before you begin implementing the best practices described in this document.

Determining Your Security Posture

The first thing you need to do is to determine your security posture. This posture applies not only to your webMethods Integration Platform, but also to all computer systems in your environment. The following are general characterizations; select the one that is closest to your organization's security posture. Determining your security posture will guide you through the rest of this document—it will help you decide which of the security best practices to follow.

Because these are generalizations, you may want to consider additional items above the level you select, or discount items at the level you select. For example, if your most critical concern is tracking data to its origin, you may want to follow the “very high” posture for digital signatures, but “medium” or “low” postures for other areas.

Very High:

- One or more of your webMethods Integration Servers are connected to the Internet, and your organization has strong policies about security configurations.
- You are concerned about attacks against your webMethods Integration Servers and Brokers from both inside and outside your organization.
- As a policy, you want to minimize the risks of accidentally malfunctioning software.
- By default, you want to disallow all access not specifically allowed.
- You plan to use HTTPS or FTP over SSL (also known as FTPS) for nearly all of your webMethods Integration Server communications, and you will use 1024– or 2048–bit public keys with 128–bit private keys for maximum SSL protection.
- You plan to use SSL for communications between your webMethods Brokers and Integration Servers.
- You plan to avoid using FTP, HTTP, and mail for webMethods Integration Server communications.
- You plan to use PKI profiles stored on an HSM device to contain keying material needed for encrypting, decrypting, verifying, and signing documents. A PKI profile contains your private key, user certificate, digital signature, CA certificate, certificate histories, and other information.
- Your network configuration includes filtering routers, firewalls, and a Demilitarized Zone (DMZ), with one or more proxy servers in front of any webMethods Integration Server or application servers, or you can use an Integration Server in a Reverse Invoke configuration in front of your internal Integration Server.
You recognize significant risks to your services, and are willing to put in the amount of effort required to maximize the security of all of your systems.

High:

- One or more of your webMethods Integration Servers are connected to the Internet, and your organization has policies about security configurations.
- You are concerned about attacks against your webMethods Integration Servers and Brokers, mostly from outside your organization, and to a lesser extent, from inside your organization.
- By default, you want to disallow all access not specifically allowed.
- Your network configuration includes filtering routers or firewalls and a Demilitarized Zone (DMZ), with one or more proxy servers in front of any webMethods Integration Server or application servers.
- You plan to use HTTPS or FTP over SSL (also known as FTPS) for nearly all of your webMethods Integration Server communications, and you will use 1024– or 2048–bit public keys with 128–bit private keys for maximum SSL protection.
- You are likely to use SSL for communications between webMethods Integration Servers and Brokers, at least for administrative actions.
- You plan to avoid using FTP, HTTP, and mail for webMethods Integration Server communications.
- You plan to use PKI profiles in your file system to store keying material needed for encrypting, decrypting, verifying, and signing documents. A PKI profile contains your private key, user certificate, digital signature, CA certificate, certificate histories, and other information.
- You perceive significant risks to your services, and are willing to put in some extra effort to ensure the security of all of your systems.

Medium:

- Your webMethods Integration Server is connected to the Internet, and your organization might have some policies about security configurations.
- You are concerned about attacks against your webMethods Integration Servers and Brokers from outside your organization, but not from inside your organization.
- You might want to allow all access not specifically denied.
- Your network configuration includes filtering routers or firewalls and may have a Demilitarized Zone (DMZ).
- Because internal attacks are not a concern, you do not plan to use SSL between and Integration Servers and Brokers.
You plan to use HTTPS or FTP over SSL (also known as FTPS) for some of your webMethods Integration Server communications, and you will use 1024–bit public keys with 128–bit private keys for maximum SSL protection.

If you use HTTP, FTP, or mail for some webMethods Integration Server communications, it is restricted to non–sensitive data or it is protected using a Virtual Private Network (VPN).

You perceive moderate risks to your services, and are willing to put in a moderate amount of effort to ensure the security of your webMethods Integration Server.

**Low:**

- Your webMethods Integration Server is in a test environment, or if it is connected to the Internet, contains no critical information.
- Your organization might some have policies about security configurations.
- You are not concerned about attacks against your webMethods Integration Servers and Brokers from inside your organization.
- By default, you want to allow all access not specifically denied.
- You will use HTTPS, HTTP, FTPS, or FTP, or mail for your webMethods Integration Server communications, and are relatively unconcerned with protecting data in transit.
- Your network configuration includes a firewall, but probably does not have a DMZ, or your system might not be connected to the Internet.
- You perceive little risk to this system, and want to minimize the extra effort required to ensure the security of your webMethods Integration Server.

**What You Need Before You Start**

Before you start reading the security best practices in this document, you need to obtain the following information:

- A list of the names and versions of all webMethods adapters and layered products in use at your site. You will also need the corresponding Best Practices documents for adapters and layered products (available through webMethods Technical Services).
- A list of all application–specific services in your webMethods Integration Server, and whether they are directly callable or for internal use within the server. For directly callable services, determine who should be able to access them (for example, administrators or trading partners).
Overview of Best Practices

This section summarizes the best practices and indicates which best practices you should follow, depending on your security posture, as described in the previous section. The first column of the table includes one through four padlock symbols, to be interpreted as follows:

- Implement if your security posture is very high.
- Implement if your security posture is high or very high.
- Implement if your security posture is medium, high, or very high.
- Implement if your security posture is low, medium, high, or very high.

These are general suggestions—your organization may differ in the importance given to particular measures. As you review this list, you may want to check the boxes (☐) you plan to implement. When you finish updating the server to meet the best practices, check this list again to make sure you have performed all of the checked steps.

<table>
<thead>
<tr>
<th>Security Posture</th>
<th>Security Measure</th>
<th>See Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐ 1</td>
<td>Sign up for the security alerts service so you will be notified of any changes to this document, or any other security-related aspects of webMethods products.</td>
<td>81</td>
</tr>
<tr>
<td>☐ 2</td>
<td>Apply these best practices to the webMethods Integration Server and Broker in your test environment before applying to a production server and Broker.</td>
<td>68</td>
</tr>
<tr>
<td>☐ 3</td>
<td>Make a backup of the entire webMethods file system before applying these best practices. You may need this backup in case of later configuration errors.</td>
<td>N/A</td>
</tr>
<tr>
<td>☐ 4</td>
<td>Using Basic Infrastructure Protections</td>
<td>19</td>
</tr>
<tr>
<td>☐ 4.1</td>
<td>Examine the placement of your webMethods Integration Server(s) within your Intranet and/or DMZ.</td>
<td>19</td>
</tr>
<tr>
<td>☐ 4.2</td>
<td>If you use a webMethods Reverse Invoke server in your DMZ, always use the SSLSOCK protocol for transmitting data between the Reverse Invoke server and the server residing in your Intranet.</td>
<td>22</td>
</tr>
<tr>
<td>☐ 4.3</td>
<td>If you use a webMethods Reverse Invoke server in your DMZ, configure external firewalls to block access to the registration listener on the Reverse Invoke server.</td>
<td>22</td>
</tr>
<tr>
<td>☐ 4.4</td>
<td>Check for security-related patches or new versions of the OS and Java VM.</td>
<td>22</td>
</tr>
<tr>
<td>☐ 4.5</td>
<td>Use firewall and router rules to allow access to only those ports used by the webMethods Integration Server.</td>
<td>22</td>
</tr>
<tr>
<td>☐ 4.6</td>
<td>Do not install the webMethods Integration Server on the same computer as the firewall.</td>
<td>22</td>
</tr>
<tr>
<td>Security Posture</td>
<td>Security Measure</td>
<td>See Page</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>☑️ ☑️ ☑️</td>
<td>4.7 Set the <code>watt.net.ssl.client.strongcipheronly</code> and <code>watt.net.ssl.server.strongcipheronly</code> parameters to true to disable use of weak cryptographic ciphers.</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>☑️ ☑️ 5 Selecting a Secure Transmission Protocol</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>☑️ ☑️ ☑️ 5.1 If possible, use HTTPS or FTPS, or consider a VPN with any other protocol.</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>☑️ ☑️ 5.2 If internal access is a concern, use the firewall to constrain it.</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>☑️ 5.3 Use SSL or a VPN for communications between the Broker and your Integration Servers.</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>☑️ ☑️ ☑️ 5.4 (For UNIX) If you want to use default port numbers, use port remapping to avoid running as root, or consider using a higher numbered port.</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>☑️ ☑️ ☑️ 5.5 If possible, run the webMethods Integration Server as a user ID with permissions to access its own files, but no other files on the system.</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>☑️ 6 Master Passwords</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>☑️ 6.1 Change the master password, which is used for encrypting outbound passwords.</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>☑️ ☑️ ☑️ 6.2 Set the Integration Server to prompt for the password at server startup.</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>☑️ 7 Authentication Mechanisms</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>☑️ ☑️ 7.1 Create new users to replace the built-in Administrator, Developer, Replicator, Central, and Broker accounts and disable the built-in accounts.</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>☑️ ☑️ 7.2 New users that are members of the Developers group should set secure passwords themselves.</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>☑️ ☑️ 7.3 For all other users, the Administrator should select secure passwords.</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>☑️ ☑️ ☑️ 7.4 Create individual accounts for all users, including Administrators and Developers (rather than sharing the predefined accounts among multiple users).</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>☑️ ☑️ ☑️ 7.5 If using locally stored user passwords (that is, not LDAP), use OS permissions to protect the password file.</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>☑️ 7.6 Although the Administrator can set minimum password characteristics, do not set them any lower than the system defaults.</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>☑️ ☑️ ☑️ 7.7 When mapping digital certificates to users, assign to the least privileged IS user name possible for that user’s responsibilities.</td>
<td>33</td>
</tr>
<tr>
<td>Security Posture</td>
<td>Security Measure</td>
<td>See Page</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>□</td>
<td><strong>7.8</strong> Use SSL and require client certificates. Use OS permissions to protect private key files.</td>
<td>33</td>
</tr>
<tr>
<td>□</td>
<td><strong>7.9</strong> Verify that <code>watt.security.ssl.ignoreExpiredChains</code> is set to “false” (default value). If this setting is not in the configuration file, add it.</td>
<td>33</td>
</tr>
<tr>
<td>□</td>
<td><strong>8</strong> Controlling by IP Address or Domain</td>
<td>37</td>
</tr>
<tr>
<td>□</td>
<td><strong>8.1</strong> Configure both the firewall and the webMethods Integration Server settings to restrict access to your trading partners.</td>
<td>38</td>
</tr>
<tr>
<td>□</td>
<td><strong>8.2</strong> Use the Allowed Hosts list to allow access only to those hosts of your partners that should be connecting to your webMethods Integration Server.</td>
<td>38</td>
</tr>
<tr>
<td>□</td>
<td><strong>8.3</strong> If possible, use IP addresses (harder to spoof) and/or configure your firewall to detect and reject domain name spoofing attempts.</td>
<td>39</td>
</tr>
<tr>
<td>□</td>
<td><strong>8.4</strong> When using one webMethods Integration Server in your DMZ and one in your intranet, configure the intranet server to only allow access from the server in the DMZ.</td>
<td>38</td>
</tr>
<tr>
<td>□</td>
<td><strong>9</strong> Review and set ACLs.</td>
<td>39</td>
</tr>
<tr>
<td>□</td>
<td><strong>9.1</strong> Review all of your custom services/folders and apply the appropriate Execute ACL: Administrators, Developers, Default, Anonymous, or custom.</td>
<td>44</td>
</tr>
<tr>
<td>□</td>
<td><strong>9.2</strong> Review all of your custom services/folders and apply the appropriate List, Read, and Write ACLs.</td>
<td>44</td>
</tr>
<tr>
<td>□</td>
<td><strong>10</strong> Set up an Administrator port.</td>
<td>45</td>
</tr>
<tr>
<td>□</td>
<td><strong>11</strong> Set up a Developers port.</td>
<td>47</td>
</tr>
<tr>
<td>□</td>
<td><strong>11.1</strong> Disable the Developer user.</td>
<td>47</td>
</tr>
<tr>
<td>□</td>
<td><strong>12</strong> Set up a Replicators port.</td>
<td>52</td>
</tr>
<tr>
<td>□</td>
<td><strong>12.1</strong> If using replication, disable the built-in Replicator user. Publishers and subscribers can have their own user names.</td>
<td>51</td>
</tr>
<tr>
<td>□</td>
<td><strong>12.2</strong> If not using replication (e.g., because you only have one Integration Server), disable or delete the built-in Replicator user.</td>
<td>52</td>
</tr>
<tr>
<td>□</td>
<td><strong>12.3</strong> Use SSL for package distribution.</td>
<td>52</td>
</tr>
<tr>
<td>□</td>
<td><strong>13</strong> Protect Internal Services from External Access (Port Controls), for each port if possible.</td>
<td>53</td>
</tr>
<tr>
<td>□</td>
<td><strong>13.1</strong> Apply appropriate firewall and IP/Domain protection.</td>
<td>37</td>
</tr>
<tr>
<td>□</td>
<td><strong>13.2</strong> Identify which services/folders should be available for external access.</td>
<td>53</td>
</tr>
<tr>
<td>Security Posture</td>
<td>Security Measure</td>
<td>See Page</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>☐ ☐ ☐ 13.3</td>
<td>Add the list of external services needed by adapters and layered products to the port’s Allow list. (Refer to the Best Practices documents for the adapters and other webMethods products.) Enter service/folder names in the form folder1.folder2.folder3:service.</td>
<td>54</td>
</tr>
<tr>
<td>☐ ☐ 13.4</td>
<td>Add your list of application–specific services that are accessible from each port.</td>
<td>54</td>
</tr>
<tr>
<td>☐ ☐ ☐ 13.5</td>
<td>Update the IP addresses for the port, specifying allowed only hosts if possible.</td>
<td>37</td>
</tr>
<tr>
<td>☐ ☐ ☐ 13.6</td>
<td>Set the list of allowed addresses for the diagnostic port to “localhost.”</td>
<td>26</td>
</tr>
<tr>
<td>☐ ☐ ☐ 14</td>
<td>Protect DSPs and other files in the pub directories.</td>
<td>55</td>
</tr>
<tr>
<td>☐ ☐ ☐ ☐ 14.1</td>
<td>Ensure that the ( \text{pub} ) directories of all packages, especially those you created yourself, are protected by a \text{access} file. Ensure that all subdirectories of ( \text{pub} ) are also protected by a \text{access} file.</td>
<td>56</td>
</tr>
<tr>
<td>☐ ☐ ☐ ☐ 14.2</td>
<td>Either set the \text{watt.server.displayDirectories} parameter to “false” or create an \text{index.html} file for every ( \text{pub} ) directory.</td>
<td>57</td>
</tr>
<tr>
<td>☐ ☐ ☐ ☐ 14.3</td>
<td>Reload the package for \text{access} file changes to take effect. Restart the server if you made changes to \text{access} files in the WmRoot package.</td>
<td>57</td>
</tr>
<tr>
<td>☐ ☐ ☐ ☐ 15</td>
<td>Special Considerations for Clustering</td>
<td>65</td>
</tr>
<tr>
<td>☐ ☐ ☐ ☐ 15.1</td>
<td>Do not run the repository server as a privileged user identity.</td>
<td>66</td>
</tr>
<tr>
<td>☐ ☐ ☐ ☐ 15.2</td>
<td>Configure your firewall to block access to the repository from the outside.</td>
<td>66</td>
</tr>
<tr>
<td>☐ ☐ ☐ ☐ 15.3</td>
<td>Use an additional firewall to protect the repository from internal attacks.</td>
<td>66</td>
</tr>
<tr>
<td>☐ ☐ ☐ ☐ 16</td>
<td>Transition from Testing to Production</td>
<td>68</td>
</tr>
<tr>
<td>☐ ☐ ☐ ☐ 16.1</td>
<td>For a production system, disable Developer access by removing all groups from the Developers ACL or removing as many users as possible from the Developers group. In addition, disable Internal access by removing all groups from the Internal ACL or removing as many users as possible from the Internal group.</td>
<td>68</td>
</tr>
<tr>
<td>☐ ☐ ☐ ☐ 16.2</td>
<td>Delete or disable sample packages provided as part of adapters and other products. Deleting is preferable to disabling.</td>
<td>69</td>
</tr>
<tr>
<td>☐ ☐ ☐ ☐ 16.3</td>
<td>Use the Security Configuration Checker to check your server configuration and point out possible security problems.</td>
<td>69</td>
</tr>
<tr>
<td>☐ ☐ ☐ ☐ 17</td>
<td>Auditing and Forensics</td>
<td>76</td>
</tr>
<tr>
<td>☐ ☐ ☐ ☐ 17.1</td>
<td>Archive audit logs by backing them up and storing them off line.</td>
<td>76</td>
</tr>
<tr>
<td>Security Posture</td>
<td>Security Measure</td>
<td>See Page</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------</td>
<td>----------</td>
</tr>
<tr>
<td>☑ ☑ ☑ ☑</td>
<td>17.2 Before going into production, calculate cryptographic checksums of all files that should not change. Check for modifications periodically.</td>
<td>76</td>
</tr>
<tr>
<td>☑ ☑ ☑ ☑</td>
<td>17.3 Before going into production, make a complete backup of the webMethods directory and store off line. Make a new backup whenever the configuration changes.</td>
<td>76</td>
</tr>
<tr>
<td>☑ ☑ ☑ ☑</td>
<td>17.4 Create and review audit logs regularly.</td>
<td>77</td>
</tr>
<tr>
<td>☑ ☑ ☑ ☑</td>
<td>18 Test the new configuration, especially using different user IDs. Check for access to the proper services on each port and for unauthenticated access.</td>
<td>68</td>
</tr>
<tr>
<td>☑ ☑ ☑ ☑</td>
<td>19 Propagate changes to other webMethods Integration Servers in your environment.</td>
<td>70</td>
</tr>
</tbody>
</table>
Using Basic Infrastructure Protections

This section describes typical network architecture for B2B and EAI environments. If you have both B2B and EAI integrations, you should consider both types of infrastructure protections.

The Integration Server is intended to be outward facing. Never directly connect a Broker to the outside world or put it in your DMZ.

The webMethods Integration Platform is part of your total Information Technology infrastructure. As such, security for the server is affected by your overall system architecture.

B2B Network Architecture

In a B2B environment, a webMethods Integration Server can be placed in your Demilitarized Zone (DMZ), in your Intranet, or both. In addition, you can place a third party proxy server, such as the IPlanet Proxy Server, in the DMZ and run it as a reverse proxy.

These options result in a number of possible configurations, as shown in the following table.

<table>
<thead>
<tr>
<th>DMZ contains...</th>
<th>Intranet contains...</th>
<th>What you get...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration Server directly connected to internal systems</td>
<td></td>
<td>Server protected by the outer firewall only.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Requires inward–facing hole in inner firewall for access to internal systems.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Because it is written in Java, the Integration server is effectively immune to the most common attacks, i.e., buffer overruns and format confusion attacks.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The DMZ administrator must know webMethods administration.</td>
</tr>
<tr>
<td>Integration Server</td>
<td>Internal server protected by the outer and inner firewalls.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Requires a hole in the inner firewall and straight–through routing of requests from the Internet to the internal Integration Server.</td>
<td></td>
</tr>
<tr>
<td>DMZ contains...</td>
<td>Intranet contains...</td>
<td>What you get...</td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Integration Server acting as a reverse proxy | Integration Server | ■ Internal server protected by the outer and inner firewalls and by the Integration Server in the DMZ.  
■ Requires an inward–facing hole in the inner firewall for access to the Integration Server.  
■ Because it is written in Java, the internal Integration Server is effectively immune to the most common attacks, i.e., buffer overruns and format confusion attacks.  
■ The DMZ administrator must know webMethods administration. |
| Third-party proxy server acting as a reverse proxy | Integration Server | ■ Internal server protected by the outer and inner firewalls and by the third–party proxy server in the DMZ.  
■ Requires an inward–facing hole in the inner firewall for access to the Integration Server. |
| Integration Server acting as a Reverse Invoke server | Integration Server | ■ Internal server protected by the outer and inner firewalls and by the Reverse Invoke server in the DMZ.  
■ Requires no inward–facing hole in inner firewall.  
■ For additional security, the internal server initiates connections to the Reverse Invoke server in the DMZ, rather than the other way around.  
■ Typically, authentication checking is performed in the internal network without custom coding. The list of usernames/passwords or acceptable certificates (or the LDAP server) does not need to be accessible from the DMZ. |

**Note:** Authentication can be performed on the Reverse Invoke server; however, this is not the recommended configuration, because it requires you to place user information and certificates in the DMZ, where they are not as safe. See “Performing Client Authentication on the Reverse Invoke Server” in the *webMethods Integration Server Administrator’s Guide* for more information.

■ Reverse Invoke uses a proprietary protocol—this is an advantage if you want to switch to a different protocol in the DMZ rather than using a standard protocol throughout.
The Integration Server in the DMZ is less vulnerable to “script kiddie” attacks than commercial reverse proxy server because attackers are less familiar with Reverse Invoke than with commercial reverse proxy servers. However, security that relies on attacker ignorance is not a strong defense, and should not be relied on as a primary security mechanism.

Using SSL involves significant overhead because the server must re-establish a new connection each time the server receives a new request from the client. With Reverse Invoke, you avoid this overhead because the connections between the Reverse Invoke server in the DMZ and the internal server are persistent.

Because it is written in Java, the Reverse Invoke server is effectively immune to the most common attacks, i.e., buffer overruns and format confusion attacks.

Some considerations:

- The DMZ administrator must know webMethods administration.
- The internal server controls the number of connections, possibly resulting in a bottleneck.
- Reverse Invoke uses a proprietary protocol—a possible drawback if you value sticking with a standard protocol rather than switching to a different one in the DMZ.
- By default, the Reverse Invoke server does not check to see whether the service the client is invoking is available on the internal server. It simply passes the request to the internal server. You can, however, configure your Reverse Invoke to check service availability before sending requests to the internal server. See “Specifying Allowed Services from the Reverse Invoke Proxy Port” on page 23 for more information.
- By default, external clients can execute DSPs and other non-involve directives on the internal server. This means that an external client could access the Server Administrator for your internal server. You can configure your Reverse Invoke server to prevent it from passing non-involve requests to the internal server. See “Preventing External Access to DSPs on the Internal Server” on page 25 for more information.

(For more information about the trade-offs among the different placements, see the webMethods 6 Security White Paper, available from webMethods Technical Services.)
Best Practice: Place the webMethods Integration Server in your Intranet, and use a webMethods Integration Server or a proxy server in the DMZ to perform initial filtering of requests. Never place a webMethods Integration Server outside your outer firewall.

Best Practice: If you use a webMethods Reverse Invoke server in your DMZ, always use the SSLSOCK protocol for transmitting data between the Reverse Invoke server and the internal server.

Best Practice: If you use a webMethods Reverse Invoke server in your DMZ, configure external firewalls to block access to the Registration Listener on the Reverse Invoke server.

A webMethods Integration Server is only as secure as the operating system it runs on. As part of securing your server, verify that the underlying operating system (such as Windows, Solaris, or HP–UX) is configured securely, and that you have installed all security patches. You should also install all patches for your Java Virtual Machine (JVM), because flaws in the JVM can lead to security problems for the server.

Best Practice: Check regularly with your operating system and JVM vendors for any patches, and install them promptly. Turn off all unneeded network services, such as TELNET servers. Use security assessment tools to determine whether there are any unpatched vulnerabilities. Tools in this category include the commercial ISS Internet Scanner (www.iss.net) and open source Nessus toolkit (www.nessus.org). There are also useful tools for verifying the correct configuration of your operating system; YASSP (www.yassp.org) is a freeware tool for Solaris systems.

Best Practice: Regardless of where you place your webMethods Integration Server, firewalls and filtering routers are a key part of protecting the server. Be sure that you allow access to only those ports that the server is actually configured to use.

Best Practice: Do not install the webMethods Integration Server on the same machine that serves as a firewall—neither is intended to be used in that way. This configuration increases the risks associated with any malfunction of the server and reduces the performance of the combined system.

When installing your webMethods Integration Server, consider the TCP/IP ports it will use and how that relates to the rest of your infrastructure. The following section discusses the trade-offs in selecting and setting up ports.
Additional Considerations for Reverse Invoke

Users of Reverse Invoke should be aware of several special considerations.

If you are using the SSLSOCK protocol to provide SSL–based communications between the Reverse Invoke server and the internal server, the internal server can prove its identity to the Reverse Invoke server using either username and password or certificate authentication. Using certificate based authentication is more secure.

Best Practice: Use certificate–based authentication with SSLSOCK, if possible.

Specifying Allowed Services from the Reverse Invoke Proxy Port

By default, the Reverse Invoke server does not check to see what service a client is requesting to execute. It simply passes the request to the internal server, which verifies that the requested service is available for execution. However, if you want the added protection of having the Reverse Invoke server verify service availability, you can set up a filtering service on the Reverse Invoke server to do so. This service contains a list of allowed services. If the requested service is on the list, the Reverse Invoke server passes the request to the internal server, which will also perform this check. If the service is not on the list in the filtering service, the Reverse Invoke server rejects the request.

Note: Typically you use a filtering service to perform additional user–defined processing, such as XML validation of requests, before passing them to the internal server, but you can use one to pre–screen for service availability on the internal server as described above.

Important! The Access Mode setting for the proxy port applies only to the filtering service that runs on the Reverse Invoke server. In fact, in order for the Reverse Invoke server to execute the filtering service, you must ensure that the filtering service is listed in the List of Allowed Services for the proxy port.
Consider a Reverse Invoke server that has two proxy ports, 1000 and 2000. You want to allow different services from each port. The following set of steps (and filter service) will allow you to enforce these restrictions:

1. Create a service, similar to the following, substituting the appropriate service names in the case 1000 and case 2000 blocks:

   ```java
   // === Begin service filter:serviceByPort ===
   int port = com.wm.app.b2b.server.ServerAPI.getCurrentPort();
   String svc = null;
   com.wm.net.HttpHeader head = Service.getHttpRequestHeader();
   if(head != null) {
       svc = head.getRequestUrl();
       if(svc != null & svc.startsWith("/invoke/")) {
           int idx = svc.indexOf('?');
           // strip off invoke and any args and convert to NS form
           if(idx != -1) svc = svc.substring(8, idx).replace('/', ':');
           else svc = svc.substring(8).replace('/', ':');
       }
   }
   if(svc != null) switch(port) {
       case 1000:
           if(svc.equals("wm.server:ping")) return; // allowed
           break;
       case 2000:
           if(svc.equals("pub.flow:tracePipeline")) return; // allowed
           if(svc.equals("wm.server:ping")) return; // allowed
           break;
   }
   // We didn't break out in the switch, so no valid service was found.
   throw new ServiceException("Can't execute "+svc+" on port "+port);
   // === End service filter:serviceByPort =====

2. Configure the Reverse Invoke server to use the filtering service. This process consists of adding parameters to the server.cnf file and adding the filtering service to the Allow List for the proxy port. See “Setting Up the Filtering Service” in the “Managing Server Security” chapter in the `webMethods Integration Server Administrator’s Guide` for instructions.

3. Verify that the Allow List for the internal server matches the Allow List you specify in the filtering service. If the two lists are not synchronized, requests might be rejected or allowed in ways you did not intend. For example, a request might get through the Reverse Invoke server, but then be rejected by the internal server. Or, a request will be stopped at the Reverse Invoke server when in fact it should have gone on to the internal server.
Preventing External Access to DSPs on the Internal Server

By default, external clients can execute DSPs and other non–invoke directives on the internal server. This means that an external client could execute the Server Administrator for your internal server. You can disable this ability by specifying the following property in the server.cnf file on the Reverse Invoke server:

```
watt.server.allowDirective=port1, port2, ..., portn
```

The variables `port1, port2, ..., portn` are port numbers of proxy ports on the Reverse Invoke server.

With this property in effect, if the Reverse Invoke server receives a request to execute a DSP or non–invoke directive through any of the specified ports, the server rejects the request and does not pass it to the internal server. The ports are still able to accept requests to invoke services from browsers or client code.

**Note:** To prevent execution of DSPs and other non–invoke directives on the Reverse Invoke server, use `.access` files. See “Assigning ACLs to Files the Server Can Serve” in the “Managing Server Security” chapter of the webMethods Integration Server Administrator’s Guide.

**Note:** Reverse Invoke can forward requests that are received via HTTP and HTTP/S only. If your installation uses FTP, FTPS, or SMTP and you want to set up a Reverse Invoke environment, you can write services to run on the Integration Server in the DMZ. These services receive the request and resubmit it to an internal HTTP port that is configured to use Reverse Invoke.

For information about using FTP with Reverse Invoke, refer to the Security Technical Article “Using FTP and Reverse Invoke” on webMethods Advantage.

For further information, contact webMethods Professional Services.
Selecting a Secure Transmission Protocol

webMethods Integration Server supports several different transport protocols for communication between webMethods Integration Servers or between one webMethods Integration Server and other types of clients or servers: HTTP, HTTPS, FTP, FTPS, and E-mail (SMTP for outbound; POP and IMAP for inbound).

**Best Practice:** Customers concerned with security should use HTTPS or FTPS, because they provide SSL protection of the traffic. If HTTPS or FTPS is not an option, you can use a Virtual Private Network (VPN) with any of the other protocols to provide protection in transit. To minimize exposure, configure the VPN to allow only the required protocol (for example, FTP) between the ends of the communication path, rather than opening the two networks to each other. Opening the network completely makes each end vulnerable to all attacks on the other.

VPN encryption is comparable to HTTPS or FTPS encryption. However, HTTPS and FTPS have the advantages of protecting the data all the way to the webMethods Integration Server and providing application-level authentication through SSL client certificates. VPN encryption, on the other hand, has some disadvantages: the data is decrypted at the VPN device, then passed unprotected to the webMethods Integration Server; and there is no application-level authentication because the VPN does not send any authentication information to the application.

When establishing a connection using HTTPS or FTPS, a webMethods Integration Server will use the strongest encryption suite it can, preferably 128-bit encryption. If the receiving server is only capable of “weak” encryption (for example, it uses a 40-bit session key), the server will accept a weak connection unless weak ciphers are disabled by setting the `watt.net.ssl.client.strongciphersonly` and `watt.net.ssl.server.strongciphersonly` parameters to “true.”

**Best Practice:** While generating public/private key pairs and creating a Certificate Signing Request (CSR), administrators should be careful to select a 1024 bit (or greater) private key. This will ensure maximum protection for your data.

The webMethods Integration Server can use the standard port numbers for each of its protocols: 21 for FTP, 25 for e-mail, 80 for HTTP, 443 for HTTPS, 990 for FTPS, and 9999 for the diagnostic port. By default, it uses port 5555 for HTTP. There are no ports preconfigured for any other services.

The diagnostic port must use HTTP; HTTPS is not allowed. As a result, data is passed in the clear, i.e. unencrypted, which is not secure. webMethods recommends that you set the port access list to allow access from localhost only. This setting allows only clients running on the machine on which the webMethods Integration Server is running to access services through this port.
The webMethods Broker can communicate with Integration Servers using an unencrypted protocol or over SSL. As an alternative to SSL, you can use a VPN to protect communications traffic.

**Best Practice:** Customers concerned with security should use SSL or a VPN for protecting traffic between the Broker and Integration Servers.

### Running Your webMethods Integration Server on a UNIX-Based System

If your webMethods Integration Server is running on a UNIX-based system (such as Solaris, HP-UX, Linux, or AIX) and you are using port numbers below 1024, configure the JVM to run as root so that the server can access those restricted ports. You can do this by (1) starting the JVM while logged in as root or (2) starting the JVM as root from a system startup file. As an alternative, your host operating system or firewall may support “port remapping,” where external users can access the default port, and the firewall or operating system forwards the request to your server using a higher numbered port.

**Caution:** Running the JVM (and therefore the Integration Server) as root on a UNIX-based system is dangerous. It can lead to total compromise of your machine. This is because any Integration Server user with Administrator or Developer authority can easily obtain root privileges by creating a root shell. These privileges give the user total control of the machine, not just the Integration Server.

**Note:** If you run the JVM as non-root, you cannot define port numbers below 1024 for the Integration Server unless port mapping (either in the operating system or in the firewall) is available.
**Best Practice:** You can minimize the overall risk by not running the JVM as root or using port remapping, and instead exposing higher numbered ports (1024 or above) to the outside world.

**Best Practice:** If exposing higher number port numbers is not acceptable (that is, you need to use default port numbers), and your host operating system supports port remapping, use this technique instead of running the JVM as root. Running the JVM as root is riskier than using port remapping, since a flaw in the JVM could lead to remote root access.

**Best Practice:** If exposing higher port numbers is not acceptable, and your host operating system does not support port remapping, but your firewall provides this feature, use this technique instead of running the JVM as root.

**Best Practice:** If possible, run your JVM as a user ID with permissions to access the webMethods Integration Server’s files, but no other files in the system. By doing so, you reduce the risk of a flaw that could allow a remote user to run an external program. For example, rather than invoking the `server.sh` script directly from the system startup script, use the command `su isuser server.sh`, and give `isuser` access only to the JVM and webMethods Integration Server files.
The following diagram shows the different options in order of the level of security they provide:

1. **Most secure:**
   - Port number greater than 1024.
   - No need to run JVM as Root.

2. **2nd most secure:**
   - Port number less than 1024.
   - OS remapping available.
   - No need to run JVM as Root.

3. **3rd most secure:**
   - Port number less than 1024.
   - Firewall mapping available.
   - No need to run JVM as Root.

4. **Least secure:**
   - Port number less than 1024.
   - No mapping available.
   - Must run JVM as Root.

---

**Firewall**

- Port 443

---

**Figure 1. Options for Running JVM on a UNIX System**
Running Your webMethods Integration Server on a Windows System

If your webMethods Integration Server is running on a Windows system, you can use the default ports without any special configurations.

**Best Practice:** No special port permissions are required to run the webMethods Integration Server. However, it is best to run the server as an identity other than Administrator, and as a user ID with permissions to access the server’s files, but no other files in the system. By doing so, you reduce the risk of a flaw that could allow a remote user to run an external program. For example, it is not recommended that you run the server.bat script directly as part of the Windows bootup process. Instead, you could create an unprivileged user isuser that has access only to the JVM and webMethods Integration Server files, then use an automatic login as isuser, and have that user's login script start the server.

Port Selection for Brokers

The webMethods Broker requires four ports (by default 6847–6850). You can replace the first three with any other set of three sequential port numbers; the fourth is fixed. There is no particular advantage to moving to a different set of ports. However, avoid port numbers below 1024 so that root privileges are not required.

**Note:** When you install the Broker, if the installer is run as root (for UNIX systems) or Administrator (for Windows systems), the installer program will automatically configure the Broker to start every time the operating system is restarted. If you run the installer as an unprivileged user, you must manually restart the Broker whenever the operating system is restarted. To avoid restarting the Broker manually each time, you can edit the system startup files so that the Broker automatically restarts when the operating system is restarted.

Choosing the Best Ports

There is no inherent security advantage to using either standard or non–standard ports. The main caveat is to avoid running the JVM as root. Attackers routinely use port scanners to look for services, so “hiding” on obscure ports is of little value. Instead, when you are selecting ports, you should focus on operational needs and ease of firewall configuration.
**Integration Server Authentication Mechanisms**

Clients or servers can prove their identity to a webMethods Integration Server by using either a user name/password combination or using SSL client certificates with a corresponding private key. For information about authenticating to a Broker, see “Authentication and Access Control” on page 61.

**User Name/Password Authentication**

The user name/password mechanism is covered in detail in the “Managing Server Security” chapter of the *webMethods Integration Server Administrator’s Guide*. Additionally, the “Using an External Directory” chapter of the *webMethods Integration Server Administrator’s Guide* describes how to use an LDAP or NIS server to store user names and passwords, rather than storing them locally on the server.

---

**Note:** The option to use an NIS server for user authentication has been deprecated in this release. webMethods recommends using LDAP for performing external authentication with the Integration Server.

---

**Best Practice:** The webMethods Integration Server comes with several built-in accounts: Administrator, Developer, Replicator, Broker, and Central. Be sure to change the passwords for all these accounts as soon as you complete the installation. These are very powerful accounts, so be sure to set secure passwords. In addition, depending on the adapters or layered product you use, there may be additional built-in accounts. You should also change these passwords immediately after installation. See the security best practices documents for adapters and layered product for further details.

---

**Note:** While the Administrator and Developer accounts (and the associated Administrators and Developers groups) have different default capabilities, they have essentially equivalent privileges. A member of the Administrators group can gain all privileges associated with the Developers account, and vice versa. You should not give users the passwords to either of these built-in accounts or assign them to either of these groups unless you can trust them to safeguard your organization's information. You should never assign trading partners to either of these groups.
Best Practice: Instead of using the built-in accounts, it is better to create an individual account for each administrator and developer, and then disable the built-in account. Advantages of individual accounts include:

- Ease of changing passwords: There is no need for sophisticated mechanisms to distribute modified passwords.
- Ease of revoking access: An individual user’s access can be revoked by disabling or deleting the account, rather than having to notify all users of the shared account what the new password is.
- Individual accountability: Audit logs will show what actions were taken by each user, rather than by any user who has the shared password.
- Less predictable account names: If you disable the built-in accounts after creating individual accounts, an attacker will be unable to break in using any of the built-in accounts, and will have to determine both the password and user name.

Best Practice: When you create local a user account (that is, an account stored on the webMethods Integration Server and not in an LDAP or NIS database), be sure to specify a password at the time you create the account. Otherwise, there could be a time interval in which an attacker could use the account to gain access to your server. Minimize this interval by setting the password before performing any other functions.

Best Practice: If using user names and passwords, allow only members of the Developers group to change their own passwords. The administrator must select secure passwords for all other users, and developers must select secure passwords for themselves.

Note: If you choose to store your usernames/passwords in LDAP, webMethods recommends that you use SSL to protect the communication channel between the Integration Server and the LDAP server.

Note: If your webMethods Integration Server is in the DMZ and you are using LDAP or NIS to store usernames and passwords, you will also need access to the LDAP/NIS server in the DMZ. If that server is inside your inner firewall, you will need to open a port to allow access through the firewall to the LDAP/NIS server, which introduces some security risk.
**Best Practice:** The administrator can set minimum password characteristics for passwords set by developers. webMethods suggests that you use the default password requirements as a minimum. The defaults are a minimum password length of eight characters, with at least one uppercase and one lowercase letter, one numeric, and one “special” character.

You can use other authoritative sources for passwords, such as a Netegrity SiteMinder server, by writing a pluggable authentication module. Documents on webMethods Advantage provide details and sample code.

**Private Key/Certificate Authentication**

The “Managing Server Security” chapter of the *webMethods Integration Server Administrator’s Guide* describes how to set up your server to use SSL client certificates for authentication.

**Best Practice:** When setting up the mapping of SSL certificates to users, be sure to assign the user identity to the least privileged user name commensurate with the user’s responsibilities. Just because users have authenticated with a certificate does not mean they should have unlimited access to the webMethods Integration Server.

**Best Practice:** The most secure method of authentication is using SSL client certificates with the corresponding private key. Unlike passwords, the secret portion of the authentication (your private key) is never transmitted to the host to whom you are authenticating yourself. However, the private key is stored in the file system, so you should use operating system file permissions to protect it. The key file should be readable only by the identity used to run the webMethods Integration Server, but not any other programs on the system.

All SSL client certificates are good for a limited period of time. The most secure setting (which is the default) is that the server will reject an expired certificate (including any signing certificate in a certificate chain). You can control this behavior by setting the `watt.security.ssl.ignoreExpiredChains` parameter in the server configuration file to “false” (the default, secure setting) or “true” (a less secure setting).

**Best Practice:** By default, the `server.cnf` file does not include an explicit setting for the `watt.security.ssl.ignoreExpiredChains` parameter. Add it to the configuration file with the value “false”.

Integration Server Access Control Mechanisms

Your webMethods Integration Server provides access to several kinds of resources:

- **Services**, which are executable code that take parameters from users, perform some actions, and (typically) generate a response. **Folders** are logical groupings of services.

- **Dynamic Server Pages (DSPs)**, which are dynamically created web pages that an Integration Server returns and displays to a user. DSPs combine presentation logic with service invocations.

- Files within the `pub` directory tree of the webMethods Integration Server's packages.

- **Java Server Pages (JSPs)**, which are dynamically created web pages that a Web server returns and displays to a user. JSP technology enables the user to combine static HTML with dynamic content specified using Java code fragments. The server automatically compiles the JSP into a Java servlet, which is executed on the server when the client requests the JSP. For security best practices for these resources see the [Web Applications Developer’s Guide](#).

The methods for controlling access to these types of resources are closely related, but not identical, because you can specify separate controls for services versus DSPs, JSPs, and files.

**Note:** The webMethods Integration Server does not support CGI scripts, NSAPI plug-ins, or ISAPI plug-ins; therefore, security issues associated with those technologies are irrelevant.

There are several levels of protection for your webMethods Integration Server.

1. **Firewalls:** Your firewalls and filtering routers prevent the attacker from outside your organization from accessing your webMethods Integration Server. Depending on your network configuration, you may also use firewalls and filtering routers to prevent attacks by insiders.

2. **SSL:** SSL prevents an attacker from reading traffic between your webMethods Integration Server and other servers.

3. **webMethods Integration Server allow/deny by IP/domain:** You can configure the webMethods Integration Server to disallow access by IP addresses or domains outside a specified list.

4. **Port controls allow/deny:** You can configure the webMethods Integration Server to prevent both authenticated and unauthenticated users from accessing internal services that are used by the externally visible services. This mechanism is known as **port controls**. For example, even if a purchase order submission accesses a database, external users should not be able to access the database directly.

5. **Access Control List (ACL) configurations:** Security features within the webMethods Integration Server can further limit the services that groups of users can access. For
example, you may have some trading partners who can query a catalog, while other partners can both query the catalog and submit purchase orders. You may place similar controls on which groups of users can access particular DSPs.

6 Application restrictions: The webMethods Integration Server application itself can enforce restrictions based on application-specific criteria. For example, your flow services can restrict the dollar value of a purchase order based on criteria in your database. These may be business rules or part of the security policy. You must implement such limitations in your webMethods Integration Server application; they are not part of the core capabilities of the webMethods Integration Server.

You can implement items 4 and 5 by dividing the set of services that the webMethods Integration Server provides into several groups:

1 Administration: Services that are needed only by administrators of the webMethods Integration Server.

2 Anonymous: Services that must be available to unauthenticated users so that they can connect, disconnect, and use guaranteed delivery services.

3 Entry point: Services that need to be available to external users. Your webMethods Integration Server might offer features such as submitting purchase orders, entering invoices, and retrieving catalog entries.

4 Internal: Services that need to be available inside the server, but should not be available to external users. These include the lower level functions used to implement the external services. For example, services to access databases or adapters to internal systems should be in this group.

5 Developer: Services that are needed to develop applications for the webMethods Integration Server, but should not be available to non-developer users and generally not accessible on a deployed server.

6 Replicator: Services that are needed to replicate packages from one webMethods Integration Server to another, but should not be available to users.
Figure 2 shows the network infrastructure of the webMethods Integration Server and how the different levels provide these protections.

<table>
<thead>
<tr>
<th>Request</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. From External Client for port 4444.</td>
<td>Denied by external firewall because it allows ports 5000–5999 only.</td>
</tr>
<tr>
<td>b. Request from external client for port 5432 on proxy server.</td>
<td>Denied by proxy server because port does not exist on proxy server.</td>
</tr>
<tr>
<td>c. Request from external client for Administrator services through port 5555 on proxy server.</td>
<td>Denied by proxy server. Administrator services not allowed from that port.</td>
</tr>
<tr>
<td>d. Request from external client for non-Administrator services through port 5555 on proxy server.</td>
<td>Denied by proxy server. Possible reasons are that the service is not on external service list, or service has an ACL that does not allow the user access.</td>
</tr>
</tbody>
</table>
The following sections describe how to implement each of these facilities, with the exception of SSL and firewall services. In addition, it describes how to set up .access files to control access to Dynamic Server Pages served by the webMethods Integration Server.

**Controlling by IP Address or Domain**

This portion of the instructions is optional, but highly recommended.

The first step in controlling access to your webMethods Integration Server is determining which IP addresses and/or Internet domains can access the server. For example, if you know that you will be communicating with company1.com and company2.com, but no others, you should start by disallowing connections from all except those domains. Similarly, if you know the IP addresses (either specific addresses or ranges) that you communicate with, you could limit connections to those addresses. Note also that you can configure this access by individual port. For example, the port for administrators can be restricted to internal IP addresses, whereas the application port can be opened to trading partner IP addresses.

You can control access by IP address or domain by using your firewall, filtering router, and/or the server itself. Consult your firewall or filtering router documentation for information about how to use those capabilities. To control access using the webMethods Integration Server, see “Controlling Access to Resources by Port” in the “Managing Server Security” chapter of the webMethods Integration Server Administrator’s Guide.

The webMethods Integration Server allows you to specify either a list of hosts (or IP addresses) to be allowed access or a set to be denied access. Setting a port type to Deny By Default allows an administrator to specify an Allowed Hosts list. Only the hosts in this list...
are allowed access via that port. Conversely, if a port is configured as Allow By Default, the administrator may specify a Denied Hosts list. In this mode all addresses except those listed are allowed.

**Best Practice:** It is safer to specify an Allowed Hosts list than a Denied Hosts list, under the security maxim of “that which is not (explicitly) allowed is denied.” However, if it is not feasible to have an Allowed Hosts list, a Denied Hosts list is better than no list at all. You might want to list domains in your Denied Hosts list that are unlikely to be legitimate trading partners, such as domains that are used for hosting home users, countries with which you do not ordinarily do business, competitors, etc. While this will not preclude an attack from these domains, it is a prudent measure and reduces the threat.

**Best Practice:** Use the Allowed Hosts list in the webMethods Integration Server (and the corresponding firewall or filtering router rules) to allow access only by those hosts of your partners that should be connecting to the server. For example, rather than allowing “*.mypartner.com”, allow “is.mypartner.com”. If you prefer to use IP addresses rather than host names, it is preferable to allow “127.10.5.27” rather than “127.10.5.*.”

**Note:** There is no implied “*” on the beginning of host names. Thus, if you list “mypartner.com” on the Allowed Hosts list and nothing on the Denied Hosts list, then host “is.mypartner.com” will not be allowed to connect.

**Best Practice:** If you use a webMethods Integration Server in your DMZ and one in your Intranet, the DMZ server should, if feasible, be configured with the Allowed Hosts list for all your trading partners, while the webMethods Integration Server in the Intranet should include only the DMZ webMethods Integration Server on its Allowed Hosts list.

**Note:** IP address and domain limitations apply only to incoming connections. They will not prevent your webMethods Integration Server from initiating a connection to hosts not included in the list.

**Best Practice:** If the set of organizations you interact with is relatively static, use IP address and/or domain limitations in your firewalls, filtering routers, and your webMethods Integration Server. While the checks may be redundant, they provide added protection in the event that the firewall is breached.

In many organizations, protection by either domain name or IP address is not a usable scheme because the IP addresses and domain names change too frequently.
Understanding Users, Groups, ACLs, Folders, Services, and DSPs

The webMethods Integration Server can control access to services, DSPs, and aliases. This section describes the methods used for each type of control. These concepts are described more fully in the `webMethods Integration Server Administrator’s Guide`; they are summarized here as a reminder of the concepts used in access controls. There are no best practices in this section.

**Controls for Services**

Figure 3 shows the relationships between users, groups, ACLs, folders, and services, and ports in webMethods Integration Server.

![Figure 3. Relationship of Users, Groups, ACLs, Folders, Services, and Ports](image)

**Best Practice:** IP addresses tend to change more than domain names, so it is more convenient to list domain names. However, it is safer to use IP addresses, since they are much harder to spoof than domain names. The trade–off will depend on your willingness to make changes compared to your aversion to risk. If possible, configure your firewalls to detect (and reject) domain name spoofing attempts.

**Note:** IP address spoofing and DNS name spoofing can limit the value of these techniques, so you should not count on them as exclusive protections. They can, however, add significant value.

**Note:** IP address and domain–based restrictions control only who can and cannot connect, not what they can do once they have connected. Even if you use Allowed Hosts or Denied Hosts lists, you must still control access to the specific services (see below).
- Users of the system represent partner organizations (for example, companies), as well as administrators and developers of the webMethods Integration Server itself.

- Users belong to groups. A user can belong to an arbitrary number of groups; groups can have arbitrary numbers of members. There are certain built-in groups, such as Administrators, Anonymous, Developers, Replicators, and Everybody. You should create other groups to represent logical groupings of your users.

- An ACL consists of two group lists: a list of groups that are allowed access, and a second list of groups that are denied access. Note that this is not allowing or denying access to the ACL itself, but rather will be used as part of the access control calculation when determining access to folders and services and other elements, such as specifications, triggers, schemas, adapter notifications and document types. Access is denied for ambiguous cases where the user is not a member of any groups listed (Allowed or Denied) or the user is a member of both a group in the Allowed list and a member of a group in the Denied list. List, Read, and Write are most often used in a developer environment.

The webMethods Integration Server comes with several predefined ACLs: Administrators, Anonymous, Developers, Internal, Replicators, CentralAdministrators, CentralUsers, BrokerAdministrators, BrokerUsers, and Default. You should create other ACLs to represent other types of access desired.

Each element can be associated with four ACLs: one List, one Read, one Write, and one Execute. The List ACL controls who can list the element. The Read ACL controls who can see an element’s source or trigger conditions. The Write ACL controls who can update or delete an element. The Execute ACL controls who can execute a service. You can specify the same ACL for the List, Read, Write, and Execute ACLs or you can specify different ones. For example, you might specify Developer as the List, Read, and Write ACLs and Default as the Execute ACL.

- Folders are collections of services and other elements. A folder may have associated ACLs, in which case the ACLs protect the folder by restricting who can access the services and other elements in the folder. Each folder can be associated with four ACLs: one List, one Read, one Write, and one Execute.

- Services may also be protected by ACLs. If a service does not have any associated ACL, then the ACLs of its parent folder (defined recursively) protects the service through inheritance. If a service has associated ACLs, then those ACLs are used to determine access, and the ACLs of any parent folders are ignored (that is, the ACL of the service overrides the ACL of the folder). You can have a service that is more or less protected than the parent folder. For example, suppose you have a folder that provides a variety of services for execution: general-purpose services, a few administrative services, and a few anonymously accessible services. By placing the Default ACL on the folder as the Execute ACL, most of the services will be protected by inheritance, while the administrative services that need more protection might have an Administrators ACL as the Execute ACL, and the anonymously accessible services might have an Anonymous ACL as the Execute ACL.
A port may provide access to one or more services. It may be configured to deny all services except those allowed (called Deny by Default) or to allow all services except those denied (called Allow by Default).

Access to elements is determined as follows (you do not need to read this list to understand the remaining instructions in this document):

- If the port does not allow the source IP address (as described above in “Controlling by IP Address or Domain”), then the server rejects the request.
- If the port is configured as Deny by Default and the requested service is not included in the list of services to be allowed, then the server rejects the request.
- If the port is configured as Allow by Default and the requested service is included in the list of services to be denied, then the server rejects the request.

**Note:** Services invoked by another service within the same webMethods Integration Server are not subject to the port constraints.

- If the user making the request has not been authenticated, then the server makes the request as the Default user.

**Note:** By default, services invoked by another service within the same webMethods Integration Server are not subject to ACL checks. You can configure ACL checks for internal service invocations on a per-service basis.

- If an element does not have any ACLs specified, the server uses its inherited ACLs; if no parent folder has any ACLs set, then the server uses the Default ACL.
- If the requested element has an ACL, then the server allows the request only if the user is a member of at least one group listed on the ACL’s Allowed groups list and is not a member of a group listed on the ACL’s Denied groups. The server denies the request in all other cases.

**Controls for DSPs and Public Files**

Your webMethods Integration Server can provide access to files stored in the file system, much as a web server does. It also includes support for Dynamic Server Pages (DSPs). The following explanation covers all files stored in the \webMethods6\IntegrationServer\packages\packagename\pub directories (where packagename is the name of the particular package, such as WmRoot). For example, the index.html file stored in each directory along with DSP files is subject to the same access rules described here.
The protection scheme for DSPs and other files stored in their directories is almost identical to that for services, with two exceptions:

- There are no port-based controls to limit which DSPs (or other files) can be accessed through a particular port.
- Access to DSPs (and other files) must be controlled for each directory. There is no inheritance or override mechanism as there is with folders and services.

Figure 4 shows the relationships for DSP protection, except that ports always serve all DSPs (you cannot restrict access to a DSP to a particular port). However, any services that a DSP invokes are subject to the access limitations of the port where the DSP was accessed.

![Figure 4. Relationship of Users, Groups, ACLs, .access Files and DSPs.](image)

See “Controls for Services” on page 39 for a description of users, groups, and ACLs. Each directory that contains DSPs may have a .access file, either shipped with a webMethods product or one that the administrator has created. If a .access file is present, it consists of lines, where each line gives the name of a DSP file in the directory (without the containing directory name) and the ACL associated with that DSP file. The .access file is always protected, so it does not require an entry for itself.

The following is an example of a .access file:

```
index.html Administrators
About.dsp Default
CheckPOStatus.dsp Partners
UpdatePOStatus.dsp Partners
```

Access to DSPs is determined as follows (you do not need read this list to follow the remaining instructions in this document):

- If the server does not allow the source IP address (as described above in “Controlling by IP Address or Domain” on page 37), then the server rejects the request.
- If the user making the request has not been authenticated, then the server processes the request as the Default user.
If the directory containing the DSP has a `.access` file and the `.access` file lists the requested DSP, then the server allows the request if the user is a member of at least one group listed on the ACL's Allowed group and is not a member of any group on the ACL's Denied groups list.

If the directory containing the DSP does not have a `.access` file, or if it has an `.access` file that does not list the requested DSP, then the server applies the Default ACL.

**Controls for Web Applications**

The Integration Server uses deployment descriptor files and ACLs to control access to servlets and JSPs that make up your Web applications. Use a deployment descriptor file to define who is authorized to access Web applications by identifying one or more Web applications and the ACL to use to control access to those Web applications. Additionally you can specify how the Integration Server authenticates a user that is accessing a Web application. By default, the Integration Server uses a basic authentication setting that invokes a browser-based screen to prompt for user name and password. You can substitute the default authentication screen with your own HTML form-based login page for one or more Web applications. For more information, see the *Web Applications Developer’s Guide*.

**Controls for Aliases**

Aliases are used to access services on remote webMethods Integration Servers. An alias consists of a name, a remote host name or IP address, a port, a remote user name and password, optional SSL settings, and an ACL.Aliases provide local users rights to use credentials registered on a remote server, and must therefore be carefully controlled.

IP or port access controls do not control access to aliases because they are only used by services on the local webMethods Integration Server (for example, `pub.remote:invoke` and `pub.remote.gd:invoke`).

The server uses the assigned ACL to determine access to aliases. A user attempting to use an alias must be a member of at least one group on the ACL's Allowed list and not a member of any group on the ACL’s Denied list.

By default, there are no aliases installed on the webMethods Integration Server.

**Setting Correct ACLs on Services**

Follow this portion of the instructions for every webMethods Integration Server.

As described above, it is important to have the proper ACLs on folders and services and other elements. Once you have completed setting up IP addresses (or domain names) that can connect to your webMethods Integration Server, the next step in securing your server is to verify that every folder, service, and other element has the correct ACL.
The default ACL settings for folders, services, and other elements that are built in to the Integration Server or that are part of adapters or layered products, such as webMethods for Trading Networks, are suitable as is and require no changes. However, you should check the ACL settings for any folders, services, or other elements that are custom built as part of your application, whether these services were built by webMethods Professional Services, a webMethods solutions provider, or your own organization.

**ACL Settings for Custom Services**

You should review every service that is unique to your application, and set an appropriate ACL. The following general guidelines may be useful:

- **Services** that are used to modify the configuration of the application, or should otherwise be restricted to administrators of your webMethods Integration Server, should generally have an Administrators Execute ACL.

- **Services** that are only used during development of your application should generally have a Developers Execute ACL. Most applications will not have any services in this category.

- **Services** that should be available to all authenticated users (regardless of who they are) should generally have the Default Execute ACL.

- **Services** intended for use by your trading partners should have an Execute ACL appropriate for the category. You may have a single group of trading partners, or you may have different sets with different levels of access to your services. If there are different sets of access, create separate groups (and separate Execute ACLs).

- **In general,** Read, Write, and List ACLs should be set for Developers only.

You should avoid any services with an Anonymous Execute ACL, because these will be accessible to any user who can connect to the port, even if they do not have a valid user ID (or client certificate). However the Anonymous ACL may be useful for selective, carefully controlled “guest” services.

**Protecting Administrative Services**

This portion of the instructions is optional.

As part of securing your webMethods Integration Server, you need to protect the administrative services. You can do this in any of three ways (from most to least secure):

1. **Use ACLs**, coupled with a customized administrator port, to control access to administrative services.

2. **Use ACLs** alone without a specific port.

3. **Rely on the firewall** to protect an open port (a port for which all services are allowed).
Each of these methods has advantages and disadvantages, and are covered in the following sections.

**Best Practice:** Using ACLs, coupled with a customized administrator port, is the best way of protecting administrative services on your server. However, this option requires the most work.

### Customized Administrator–Only Port

The most secure way to protect administrative services is to identify the specific services that should be accessible to administrators, and to create a port for those services. The result is more secure, because even if an intruder gains access to the port, services with the Default Execute ACL will not be available.

To use this option, create a new port with Deny by Default as the port type and add all services that are assigned the Administrators Execute ACL. To create a new port, follow the instructions in “Configuring Ports” in the “Configuring the Server” chapter of the *webMethods Integration Server Administrator’s Guide*. Next, follow the instructions in Appendix A, “Externally Visible Services” to easily configure the port so it makes services that are appropriate for an administrator available. For additional information, refer to “Controlling Access to Resources by Port” in the “Managing Server Security” chapter of the *webMethods Integration Server Administrator’s Guide*.

**Best Practice:** Even though the services are protected by the Administrators ACL, it is still a good idea to use your firewall or filtering router to protect this port against access by those outside your organization, as well as those inside the organization who do not need to administer the webMethods Integration Server.

**Note:** Using this approach, you should avoid listing the administrative services on the externally visible services list for any port other than the port designated for the administrator’s use.

### Using ACLs to Protect the Administrative Services

An easy alternative to setting up a separate administrator’s port is to rely on the ACLs as set up to protect the services, and not to set up a specific administrative port. This option brings moderate risk, because the administrative services are now exposed to any user who guesses the Administrator password or the password of any other user in the Administrators group (subject only to the limits on IP addresses that can connect to your server).

To use this option, no additional configuration is necessary for either the webMethods Integration Server or your firewalls.
Relying on the Firewall to Protect an Open Port

The simplest but least secure option for protecting administrative services is to create an Allow by Default port where all services are accessible, and then use your firewall or filtering router to protect against unauthorized access. This approach is not recommended, because it leaves your server vulnerable to any errors in firewall configuration.

To create a new port, follow the instructions in “Configuring Ports” in the “Configuring the Server” chapter of the *webMethods Integration Server Administrator’s Guide*. Next, set the port type to Allow by Default, following the instructions in “Controlling Access to Resources by Port” in the “Managing Server Security” chapter of the *webMethods Integration Server Administrator’s Guide*. Do not list any services in the Deny list.

**Caution:** If you choose to have an open port, use firewalls and/or filtering routers to protect the open port against access both from outside users and unauthorized inside users.

Diagnostic Port

The installation process automatically sets up a diagnostic port at 9999. By default, the port allows all hosts to connect to the server. However, users can access only the services defined with the Administrators ACL. If you add services to be used from this port, protect them with the Administrators ACL.

Protecting Developer Services from Unauthorized Access

This portion of the instructions is optional.

Every *webMethods Integration Server* requires developers to create the application. Generally, developers should not have access to the server once the application goes into production.

As part of securing your server, you need to control how developers use it. You can do this in any of the following ways (from most to least secure):

1. Use ACLs, coupled with a customized developer port, to control access to developer services.
2. Use ACLs alone without a specific port.
3. Rely on the firewall to protect an open port.

Each of these methods has advantages and disadvantages, as covered in the following sections.
Best Practice: Using ACLs, coupled with a customized developer port, is the best way of protecting developer services on your server. However, configuring this option requires the most work.

Best Practice: If you use ACLs and a customized developer port, or rely on the firewall to protect an open port, it is best to disable the developer port before moving into a production environment.

webMethods Developer supports the HTTP and HTTPS protocols (but not FTP, or certificate-based authentication) for communicating with the webMethods Integration Server. As a result, user name/password is the only form of authentication that developers can use on the port.

Customized Developer-Only Port

If you choose this option, you will create a new port that allows access to only those services used by developers, then use your firewalls to protect that port.

To create a new port, follow the instructions in “Configuring Ports” in the “Configuring the Server” chapter of the webMethods Integration Server Administrator’s Guide. Next, follow the instructions in Appendix A, “Externally Visible Services” to easily configure the port so it makes services that are appropriate for a developer available. For additional information, refer to “Controlling Access to Resources by Port” in the “Managing Server Security” chapter of the webMethods Integration Server Administrator’s Guide.

Note: To perform application development, developers need access not only to the built-in services, but also to the services they develop. If possible, group developer services into one or more folders, so that you can give access using a wildcard. For example, if the application services can be grouped into the “acme” folder, then you can make the “acme” folder accessible through the developer port. By doing this, you will not have to list each individual folder and service used by the developer.

Best Practice: Because some of your developed services will be intended for external use, and others for internal use only, it’s best to group them into separate folders. For example, “acme.internal.*” could be the internal services, while “acme.external.*” could be the externally accessible services. You can give developers the ability to invoke “acme.*” on the developer port, while giving production users the ability to invoke only “acme.external.*” on the public port. You can do this even if internal and external services are mixed in the same folder, but it is easier if they are grouped separately.
Using ACLs to Protect the Developer Services

The second alternative is to rely on the ACLs as set up to protect the services, and not to set up a specific developer port. This is a simple alternative, but brings moderate risk, because the developer services are now exposed to any user who guesses the Developer password or the password of any other user in the Developers group (subject only to the limits on IP addresses that can connect to your server).

No additional configuration is necessary for either the webMethods Integration Server or your firewalls.

Relying on the Firewall to Protect an Open Port

The simplest but least secure option for protecting developer services is to create an Allow by Default port where all services are accessible, and then use your firewall or filtering router to protect against unauthorized access. This approach is not recommended, because it leaves your server vulnerable to any errors in firewall configuration.

To create a new port, follow the instructions in “Configuring Ports” in the “Configuring the Server” chapter of the webMethods Integration Server Administrator’s Guide. Next, set the port type to Allow by Default, following the instructions in “Controlling Access to Resources by Port” in the “Managing Server Security” chapter of the webMethods Integration Server Administrator’s Guide. Do not list any services in the Deny list.

Note: If you choose to use this option, you can use the same port for developers and administrators.

Best Practice: If you choose to have an open port, use firewalls and/or filtering routers to protect the open port against access both from outside users and unauthorized inside users.
Protecting Replicator Services from Unauthorized Access

This portion of the instructions is optional.

You may use replication services to copy packages automatically from one webMethods Integration Server to another. If you have more than one server operating in a cluster, you may be using replication services. Or, if you are sending or receiving packages from a trading partner, you may be using replication services.

As part of securing your webMethods Integration Server, you need to control how replication will occur. This will depend in large part on whether you are replicating packages within your organization, or between organizations (for example, with partner servers). You can control access to replication services with the following methods:

1. Use ACLs, coupled with a customized replicator port, to control access to replication services.

2. Use ACLs alone without a specific port.

Each of these methods has advantages and disadvantages, as covered in the following sections.

**Best Practice:** Using ACLs, coupled with a customized replicator port, is the best way of protecting replication services on your server. However, configuring this option requires the most work.

**Best Practice:** HTTPS or FTPS protects your packages from modification during transit. Therefore they are the preferred transport protocols when performing replication.

Replication with earlier webMethods Integration Servers uses push technology (from a publishing server to subscribing servers). Beginning with webMethods Integration Server 4.6, pull technology is also available (subscribing servers retrieving the from publishing server).

There are two parts to package replication:

1. **Setting up the Subscription.** This part is symmetric and, if you allow, both the publisher and the subscriber can set up the subscription.

2. **Package Distribution.** This is asymmetric and you can choose whether the Publisher can “push” packages when it chooses or whether the subscriber “pulls” packages when made available by the publisher.
Figure 5 shows replication within and across organizations.

In Part (a), the Alpha and Beta organizations each perform replication internally, but do not share services with each other. Attempts by users to connect to the replicator port (as described below) may be blocked by a firewall based on IP address. If the user is allowed through the firewall (or if there is no firewall), the user may be unable to perform any replication services without authenticating as a user who is a member of a group is assigned to the Replicators Execute ACL. In this case, the publishing webMethods Integration Server does not have a replicator port at all, because replication is configured for “push” operation only (that is, from the publishing server to the subscribing servers).

In part (b), the Alpha organization distributes its packages to the Beta organization. Alpha’s firewall must be configured to allow its webMethods Integration Server to initiate a connection to Beta’s webMethods Integration Server. Beta’s firewall must be configured to allow its webMethods Integration Server to receive the connection on its replicators port. However, Beta’s firewall should not be configured to receive connections on its replicator port from Gamma’s webMethods Integration Server. Gamma’s firewall is also configured to prevent connections from Alpha’s webMethods Integration Server. Because replication is asymmetric, Alpha’s firewall may or may not allow through connections from Beta’s and Gamma’s webMethods Integration Servers. If Alpha’s firewall blocks connections from Beta and Gamma, then Alpha’s administrator must perform all of the configuration on the Alpha webMethods Integration Server. However, if Alpha’s firewall
allows incoming connections from Beta and Gamma, they can sign up directly for replication services. For large trading networks, this may be more appropriate, since it imposes less of a burden on Alpha’s administrator.

In both scenarios, however, you may configure the webMethods Integration Servers to exchange various types of documents. The limitation applies only to package replication.

**General Replication Configuration**

Before setting up a replicator port, you must configure the users and passwords to be used for replication. Throughout this discussion, the server on which the package originates is referred to as the publisher, and the recipient servers are the subscribers.

**Setting up Replication for webMethods Integration Server**

The user name and password can be different on the publisher and the subscriber. The user name is not hard coded to be Replicator. The only restriction is that the user name used for replication must belong to the Replicators group or any other group that is assigned to the Replicators Execute ACL. You **can** use SSL to set up the subscription and to distribute the package.

**Publisher subscription setup:** The publishing webMethods Integration Server needs a user name and password on the subscribing webMethods Integration Server. The user name must belong to the Replicators group or any other group that is assigned to the Replicators Execute ACL. The subscriber will have to provide this user name to the publisher. In this use case, the subscriber does not need to have information about a user name/password on the publisher’s end.

**Subscriber subscription setup:** The subscribing webMethods Integration Server must have a valid user name and password for the publishing webMethods Integration Server, and also needs to provide a user name and password to the publishing server. The publishing server will use this name when it is ready to distribute the package to the subscribing server.

**Note:** The options described here are valid if **all** servers in the replication set (that is, all publishers and all subscribers) are running webMethods B2B 3.5.1 or higher. If you are running an earlier version, see the [*webMethods B2B 3.x Security Best Practices*](#).

**Best Practice:** Delete the Replicator user from the webMethods Integration Server. In its place, create a new user that belongs to the Replicators group. This user name can be different on the publisher and subscriber webMethods Integration Servers. Using a name other than Replicator makes it harder for an attacker to guess the user name.

**Best Practice:** Set the password for the newly created replication account to something that an attacker could not guess.
If the administrator for the publishing server does not want subscribers to have the ability to set up their own subscriptions and wants to manually set up subscriptions for all subscribers, then it is not necessary to provide a user name and password with replication privilege to all subscribers. In this case, the administrator of the publishing server can receive subscription requests via e-mail or phone and enter these servers as subscribers.

**Best Practice:** Use HTTPS or FTPS (that is, an SSL–protected connection) to set up subscriptions and distribute the packages. This provides protection against theft of the software, as well as against tampering with the software in transit between the publishing and subscribing servers.

Continue with the section entitled “Customized Replicator–Only Port” or “Using ACLs to Protect the Replicator Services” (below).

**Customized Replicator–Only Port**

If you choose this option, you will create a new port that allows access to only those services used by replication, then use your firewalls to protect that port. The firewall protection may include allowing access to the port from your trading partners’ webMethods Integration Servers.

To create a new port, follow the instructions in “Configuring Ports” in the “Configuring the Server” chapter of the *webMethods Integration Server Administrator’s Guide*. Next, follow the instructions in Appendix A, “Externally Visible Services” to easily configure the port so it makes services that are appropriate for a replicator available. For additional information, refer to “Controlling Access to Resources by Port” in the “Managing Server Security” chapter of the *webMethods Integration Server Administrator’s Guide*.

**Note:** Using this approach, you should avoid listing the replicator services on the access list for any port other than the port designated for replication use.

**Using ACLs to Protect the Replicator Services**

The second alternative is to rely on the ACLs as set up to protect the services, and not to set up a specific developer port. This is a simple alternative, but brings moderate risk, because the replicator services are now exposed to any user who guesses the Replicator password or the password of any other user in the Replicators group (subject only to the limits on IP addresses that can connect to your server).

No additional configuration is necessary for either the webMethods Integration Server or your firewalls to use this option.
Protecting Internal Services from External Access (Port Controls)

Follow this portion of the instructions for every webMethods Integration Server.

The next step in securing your webMethods Integration Server is to control access to those services needed by external users (in contrast to those services the server uses for internal functions). For example, external users need access to high-level database functions, but should not have access to services that access the database directly. The mechanism that controls which services are externally visible is known as port controls.

Best Practice: To maximize security, we recommend that you identify those services that should be externally visible, and protect all other services. This approach is preferable to starting with a complete list of services and eliminating those that should be protected, since even one missed service could introduce security risks and endanger your server.

In addition to the services expressly used by Administrators, Developers, and Replicators (as described in previous sections), the following sets of services should be externally visible:

Client services:
Services built in to the webMethods Integration Server that must be accessible to use the webMethods Client API. The server automatically adds these services to the allowed list when you create the port. These services are `wm.server:ping`, `wm.server:noop`, `wm.server:connect`, `wm.server:disconnect`, and `wm.server:getServerNodes`.

Guaranteed Delivery services:
Services that are used for the webMethods Guaranteed Delivery API. The server automatically adds these services to the allowed list when you create the port. These services are `wm.server.tx:start`, `wm.server.tx:restart`, `wm.server.tx:execute`, and `wm.server.tx:end`. If you are not using the Guaranteed Delivery API, you do not need these services.

Adapter services:
Services that are exported directly by the adapters you are using. See the Security Best Practices documents for your adapters, where you will find a list of services that must be exported for each adapter. You must enter these manually in the Integration Server Administrator using the Access Edit Mode screen for the appropriate port.

Application-specific services:
Services defined by your application that should be externally accessible. Obtain this list from the developers of your webMethods Integration Server application.

Before proceeding, be sure you have a complete list of services that should be externally visible. The following sequence will set up your server to prevent access to all except the listed externally visible services. For details about how to perform each step, see

**Best Practice:** Repeat these steps for each port except the administrative, developer, and replicator ports (set up in the previous sections). Do not perform these steps for the diagnostic port either.

1. Set the access mode for the port to Deny by Default, which is the default for new ports.

2. Add the adapter and Application–specific services that are to be available on the port.

**Note:** You might want to make some of your adapter services and application–specific services available from only particular ports. For instance, if you have services that only your preferred partners should access, make them available from port 5557 and use your firewall configuration to allow access to only the IP addresses of your preferred partners. In contrast, you could make all your publicly available services available from port 5555.

**Note:** When you set up the port controls, you will receive a warning that you may be preventing access to certain services. You may ignore this warning, provided that you have identified all services that should be externally visible.

**Caution:** When entering services or folders for a port, service/folder names should have the form `folder1.folder2.folder3:service`, where `folder1`, `folder2`, and `folder3` are the nested folders and `service` is the actual service name. Carefully check all values in the externally visible services list to ensure that you have entered them correctly.

**Note:** Make sure that you have selected the port type before proceeding with adding the service/folder names. Changing the port’s access mode resets the ports service list to the default values.

**Best Practice:** After setting up the externally available services, review the configuration for every port to verify that you have listed all services they should provide, and that the port type is set to Deny by Default.

**Best Practice:** If there are some ports that cannot be set to Deny by Default, verify that your firewalls block access to those ports. If possible, block access to those ports from the inside too, allowing through only those individuals (such as administrators) who will need to access the Allow by Default port.
Relationship of Dynamic Server Pages and Port Controls for Services

Your webMethods Integration Server can provide access to dynamic server pages (DSPs). Although access to the DSPs themselves is not limited by port controls, embedded service invocations in the DSPs are subject to port controls.

A DSP can be pure HTML, or more commonly, it includes service invocations of the form `%invoke pub.db:getTableInfo%`. If your DSPs include service invocations, each of the top–level services they invoke (in this example, `(pub.db:getTableInfo)`) must be accessible through the port where the DSP is retrieved.

For example, the `db-alias.dsp` DSP directly invokes the following services:

```
wm.server.db:dataSourceList,
wm.server.db:dataSourceAdd
wm.server.db:dataSourceChange
wm.server.db:dataSourceDelete
```

For the operation to run successfully, the user that runs the `db-alias.dsp` DSP must be able to access each of these four services on that port. For a Deny by Default port (the recommended setting), each of the four services must be on the Folder/Service List. For an Allow by Default port, none of these services (or any of the folders that contain them) should be on the Folder/Service List.

If any of these four services invokes other services, the lower level services do not need to be externally visible on the port.

Setting Up .access Files

Follow this portion of the instructions for every webMethods Integration Server.

Your webMethods Integration Server can provide access to DSPs and other files. Control of DSP files is achieved via `.access` files placed in each directory. The controls you specify via `.access` files apply to all files in the `webMethods6\IntegrationServer\packages\packagename\pub` directory, whether the file is a DSP file, an HTML file, or some other type of file.

```
Note: Requests for the .access file are always denied. You do not need to include an entry for it in the .access file for itself.
```

Unauthenticated users do not require access to DSPs to use the basic webMethods Integration Server product or any webMethods–supplied adapters. Your environment may want to make certain files available this way, such as if you provide a “welcome” message telling partners how to sign up to perform online transactions.
In addition, you should review all DSPs created as part of your application to determine appropriate settings for `.access` files.

**Note:** If the directory containing a DSP does not have a `.access` file, or if it has a `.access` file that does not list the requested DSP, then the server applies the Default Execute ACL.

**Note:** An alternative to listing each file in the `.access` file is to create a `.access` file with a wildcard, for example:

```
* Administrators
```

This is a catchall that will protect all files not otherwise associated with an Execute ACL. Attempts to retrieve the `.access` file are always denied.

**Best Practice:** Using a text editor, create or edit the `.access` file for each directory under `pub`. Be sure that every DSP present in the directory is listed in the `.access` file with the appropriate Execute ACL. Verify that there is a `.access` file for every directory, and every file in that directory has an appropriate Execute ACL listed in the `.access` file.

**Best Practice:** Do not use the Anonymous ACL unless you want to make the DSPs available to unauthenticated users.

**Best Practice:** Protect the `.access` files from modification using your operating system's file protections. For example, on UNIX systems, the user ID running the webMethods Integration Server should have read permission, but not write permission to the file. (If the webMethods Integration Server is running as root, this will not be possible, since root has permission to write to every file on the system.)

**Best Practice:** If your webMethods Integration Servers is used in a production environment, the entire `pub` directory should be configured as read only to the user ID running webMethods Integration Server.

**Caution:** If your server is running on a UNIX–based system, the listings for DSP files must be identical to the file names, including matching case. For example, if your `.access` file lists `CreateService.dsp`, but the actual name of the file is `createService.dsp`, the `.access` file entry will not provide the expected protection. If your server is running on a Windows 2000 system, case is unimportant, but it is still a good idea to match case exactly so that a future migration to a UNIX–based system would not expose your DSP files.
Just as with a web server, if a user requests access to a directory, the webMethods Integration Server will return the contents of the index.html file (if one exists), or a listing of files in the directory (if there is no index.html file).

**Best Practice:** Create an index.html file for every pub directory, or as an alternative, set the watt.server.displayDirectories parameter to “false” so that an attacker cannot access the list of files through your webMethods Integration Server. Any attempt to access the directory will result in a “404” error (file not found).

**Note:** This parameter does not provide any protection against an attacker accessing the files themselves. It only prevents the attacker from obtaining a list of files available to be retrieved.

**Best Practice:** Changes to .access files do not take effect until the containing package is reloaded. In the case of the WMRoot package, the changes do not take effect until you restart the webMethods Integration Server.
Broker Considerations

The Broker, in conjunction with the Broker Server, manages the flow of documents to and from network services. It connects components such as Integration Servers, clients, Brokers, and various applications. The Broker and Broker Server route, queue, and filter documents. This chapter provides best practices to ensure that your Broker is part of a secure integration.

In addition to this chapter you can read an overview of the Broker Server security capabilities in the *webMethods 6 Security White Paper*, available from webMethods Advantage.

In addition to this document, you should be familiar with the following chapters from the *webMethods Broker Administrator’s Guide*:

- Chapter 11: Managing Broker Security. This section describes in detail how to set up SSL, access controls on client groups, and firewalls.
- Chapter 6: Managing Client Groups. Client groups are the primary access control mechanism in Broker Server.

Basic Infrastructure Protections

webMethods Broker Server is part of your total Information Technology infrastructure. As such, security for the Broker Server and its configured adapters is affected by your overall system architecture. The following sections describe issues related to overall configuration of your Broker Server brokers and adapters.
Broker Server and Adapter Placement

Due to its nature, Broker Servers are typically placed on an internal, corporate network not accessible to the public (e.g., your Intranet).

Figure 6. Placement of Broker Servers

Similarly, adapters typically reside on the machine hosting internal corporate services such as SAP, Siebel, or Oracle.

Figure 7. Placement of Adapter

Best Practice: Place the Broker Server and adapters in your Intranet. If you need to receive requests from external business partners, use an Integration Server in your DMZ. Never place a Broker Server in your DMZ or outside your firewall.
**Firewall Configuration**

Broker Server communicates on a small list of TCP ports. The table below lists the default ports required by Broker Server and is a starting point for configuring internal routers and firewalls. This list does not include ports used by adapters to talk to their native resource. Broker Server can also be configured to send SNMP traps (outbound UDP port 162).

<table>
<thead>
<tr>
<th>Port</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>6847</td>
<td>Mutually authenticated (server to client and client to server) traffic between Integration Servers and the webMethods Broker Server, whether encrypted or not.</td>
</tr>
<tr>
<td>6848</td>
<td>Unidirectional (server to client) authenticated traffic between Integration Servers and the webMethods Broker Server, whether encrypted or not.</td>
</tr>
<tr>
<td>6849</td>
<td>Unauthenticated transmissions between Integration Servers and the webMethods Broker Server. All unauthenticated transmissions are also unencrypted.</td>
</tr>
<tr>
<td>6850</td>
<td>Monitoring the status of the webMethods Broker Server</td>
</tr>
<tr>
<td>6853</td>
<td>Adapter Monitor</td>
</tr>
</tbody>
</table>

**Best Practice:** Regardless of where you place your Broker Server, firewalls and filtering routers are a key part of protecting the server. Be sure that you only allow access to those ports that the Broker Server is configured to use. None of these ports should be accessible outside of your enterprise.

**Best Practice:** It is possible to change the default ports for broker traffic. Running applications on non–default ports is typically considered more secure.

**Host Level Protections**

Brokers and adapters require an underlying operating system on which to run. Properly configured, operating systems can provide a great deal of protection to applications. The following section describes how to use the capabilities of the host operating system to reduce the risk of the Broker Server Installation.

**Certificate Store Protection**

Broker Server stores certificates used for authentication and encryption in a password–protected keystore. Compromise of this file could compromise the entire Broker Server configuration.
Authentication and Access Control

Out of the box, the Broker Server allows any client or Integration Server to connect and configure Brokers and adapters. This section describes how the Broker Server can be configured to require the Broker administrator, adapters, and Broker clients to prove their identity to a Broker Server. Refer to the *webMethods Broker Administrator’s Guide*.

**Administrative Access**

Broker Servers are administered using the WmBrokerAdmin package running inside an Integration Server. By default, anyone with a running WmBrokerAdmin package can administer the broker server.

**Best Practice:** Configure the Broker Server for SSL and require a specific client identity to connect. Restrict access to the ‘admin’ client group. See the “Managing Broker Security” chapter of the *webMethods Broker Administrator’s Guide* for information about how to do this. If possible, run Broker Administrator from a dedicated Integration Server.

**Best Practice:** Restrict access to the services in WmBrokerAdmin.

**Locking Down Client Groups**

Client groups are used to control document publish/subscribe rights and are the primary access control mechanism in the Broker Server.

**Best Practice:** Restrict access to sensitive client groups to only those adapters/clients that require it. Put the minimum set of documents required to function in the client group.

**Best Practice:** Restrict access to the admin, accessLabelAdapter, eventLog, and adapters client group.

**Client Certificates**

Broker Server configurations consist of many client and server components. Authentication in Broker Server requires certificates for every client and server.
**Best Practice:** Create certificates for each Broker, adapter, Broker client, and administrative user. This assists in auditing as well as reduces the risk if a certificate is compromised. At a minimum, create certificates for each individual category.

**Best Practice:** Do not use administrative certificates to run adapters. Adapters must be configured using a client certificate in the admin client group, but they do not need administrative privilege to run.

**Territories and Gateways**

Broker Server provides the capability to group brokers into Territories. Brokers in the same territory share common definitions of client groups and document types. Clients connected to one broker in a territory can send messages to clients connected to any other broker in the territory. Brokers in a territory usually reside on separate computer systems. Therefore, the path between brokers in a territory may need to be protected.

The following diagram shows a territory consisting of two Brokers.

![Broker Territory Diagram]

**Figure 8. Broker Territory**

**Best Practice:** Require SSL client certificates for Broker–to–Broker communication in a territory. See “Using SSL for Territories” in *webMethods Broker Administrator’s Guide* for information about how to configure this option.
Territory gateways allow a limited set of messages to flow back and forth between different territories. Brokers in different territories can have different document and client group definitions.

**Best Practice:** Require SSL client certificates for broker-to-broker communication across territory gateways. See Using SSL across Territory Gateways in the *webMethods Broker Administrator’s Guide* for information about how to configure this option.

**Best Practice:** Gateways relay documents between territories. Each gateway maintains a list of documents that it forwards or receives to other systems. This list should be carefully reviewed to ensure only the required documents are allowed to flow across.

**Best Practice:** Set up document filters on the gateway to further restrict what flows between two territories.
Outbound Passwords

As part of its normal operations, the Integration Server connects to applications and subsystems such as remote Integration Servers, proxy servers, and databases. These applications require the Integration Server to supply a password.

By default, the Integration Server stores these outbound passwords in a file and encrypts them using Password-Based Encryption (PBE) technology, also known as PKCS#5. This encryption method requires the use of an encryption key or master password that you specify. The master password is also encrypted, and by default, is stored in a file.

When you first install the Integration Server, the master password is "manage." As you do with other default passwords, you need to change this password after installing the Integration Server.

For greater security, consider configuring the Integration Server to prompt for the master password at server startup rather than store it in a file.

For more information about configuring outbound passwords, refer to the “Managing Server Security” chapter in the webMethods Integration Server Administrator’s Guide.
Special Considerations for Integration Server Clustering

If you use webMethods Integration Server clustering to provide scalability, failover, and recovery, there are additional security considerations. Skip this section if you do not use server clustering.

Each of the security parameters described in this document must be applied to every webMethods Integration Server in the cluster. You can do this by making the changes on each server directly, or by copying key files, as described in “Propagating Changes Across webMethods Integration Servers” on page 70.

The shared repository, which holds webMethods Integration Server state information and utilization metrics for use in load balancing and automatic failover, makes multiple servers look and behave as one. The repository can reside in your file system or in a JDBC–compliant database. In addition, the repository can be local (residing on the same machine as the webMethods Integration Server) or remote (residing on a different machine than the webMethods Integration Server). Figure 9 shows the four common repository architectures. The double lines indicate RMI protocol, the dashed lines indicate a proprietary JDBC protocol between the JDBC driver and the JDBC server, and the dotted lines indicate an internal API call within the server.

![Diagram of Repository Architectures]

(a) Distributed repository using file storage  (b) Distributed repository using database storage

(c) Single host repository using file storage  (d) Single host repository using database storage

Legend:
- Internal APIs
- Proprietary JDBC
- RMI

Figure 9. Repository Architectures
As shipped, the RMI traffic between the webMethods Integration Server and the repository server is not encrypted, nor does the repository server demand any authentication before accepting a connection from the webMethods Integration Server. You may be able to obtain third-party protection for the RMI communications; webMethods does not provide such protection.

**Note:** RMI uses a defined port for access to the RMI registry, and a second dynamically allocated port for communication between the RMI client (the webMethods Integration Server) and the RMI server (the repository server). The port number used by the RMI registry is defined as part of the repository configuration. If there are any firewalls between the webMethods Integration Server and repository server, they must be capable of allowing both types of traffic through.

The traffic between the repository and the JDBC database (if one is in use) is protected using whatever mechanisms are provided by the JDBC driver and the database. webMethods does not provide any protection for this communications traffic. However, the repository server is configured to provide a user name and password to access the JDBC database.

There is no significant difference in the security of the repository whether using file-based storage or a JDBC-compliant database.

**Best Practice:** The repository server runs as a program within a JVM. Start the JVM as an unprivileged user, so the repository server runs as with an unprivileged identity. For UNIX-based systems, it is important for security reasons that you do not use a low numbered port (less than 1024).

**Best Practice:** If you are using the distributed approach (Figure 9(c) or (d) above), select the ports that will be used for communications from the webMethods Integration Servers to the repository server. Then, using a firewall or filtering router, block access to these ports by any hosts except the webMethods Integration Servers, particularly if using file system repository storage.

**Best Practice:** Use an internal firewall to protect your repository from internal attackers.
Figure 10(a) shows a potential configuration that protects the repository from outside attacks, while Figure 10(b) shows a configuration that protects the repository from both inside and outside attacks. Using an internal firewall protects your repository from attack by internal attackers, presuming it has been configured to block access to the port used by the repository.

(a) Protection Against External Attacks

(b) Protection Against Internal & External Attacks

Figure 10. Protecting the Repository
Transitioning from Testing to Production

In many environments, looser controls are enforced when a system is in a testing environment compared to a production environment. Before you transition a system, it is important to verify that you have followed all recommendations in this Best Practices document for the production server, including:

- Setting the passwords for all privileged accounts
- Changing the master password for outbound passwords so that the master password on the production system is different from the master password on the test system.
- Changing from password–based authentication to client certificate based authentication, if feasible
- Setting port access lists and ACLs appropriately for the Integration Server
- Setting client groups appropriately for Brokers
- Creating users with the minimum privilege possible, and verifying that non–administrative users are not members of the Administrators or Developers groups
- Verifying that you have installed all operating system and JVM patches
- Verifying that firewalls provide adequate protection from attacks
- Verifying that all ports are set to Deny by Default so non–listed services will be inaccessible. If this is not feasible, verify that your firewall prevents access to the ports that are set to Allow by Default.
- Verifying that replication services are protected from outsiders, both by verifying membership in the Replicators group and by checking access to the replicators–only port (if you created one).

**Best Practice:** Do not use the same password for the Administrator account on your production systems as you do on your development systems.

**Best Practice:** webMethods recommends disabling all developer access to the production system, both to prevent accidental changes to the configuration and to avoid providing attackers with a means of gaining privileged access to the webMethods Integration Server. Remove all groups from the list of Allowed Groups on the Developers ACL. If this is not feasible, you should remove as many users as possible from the Developers group.

If you set up a developers–only port, consider disabling that port before going into production.
Checking Your Configuration

To assist you in testing the security of your configuration, webMethods provides a configuration checker, referred to as the Security Configuration Checker (SCC). This utility, available on webMethods Advantage, compares your Integration Server’s configuration against a Best Practices configuration. The utility produces a report that points out potential problems and makes recommendations on how to avoid them.

Run this utility before putting a development system into production. It is also a good idea to run the utility periodically against your production system, in case any changes to the system have inadvertently affected security.

You should also consider running the utility early in the development process to prevent unauthorized users from accessing your development system. Because looser controls are often in effect in a testing environment, you might choose to ignore some of the recommendations until you go into production.

Note: The SCC does not detect security weaknesses in the underlying operating system, Java Virtual Machine, backend systems, or custom applications.

For more information about his utility, refer to Integration Server Security Configuration Checker Guide, available on webMethods Advantage.
Propagating Changes Across webMethods Integration Servers

The instructions in this document have thus far described how to make the changes for a single webMethods Integration Server. While you can copy packages automatically between servers in a cluster, configuration information, such as users, groups, ACLs, and ACL settings, are not automatically duplicated.

You can deploy new servers either for production, testing, or clustering by copying configuration files from one server to another. We recommend copying entire files rather than using the webMethods Integration Server Administrator to set each attribute, which could result in errors and a less secure environment.

Best Practice: When copying configuration files between systems, use a secure transfer, such as an encrypted transport or physically protected media, or a secure network.

1. Install the new server according to the standard installation procedure documented in the webMethods Installation Guide.

2. Copy the files shown in the table below from the server that is configured correctly to the server that you want to have an identical configuration. The directories are expressed relative to the home directory of the webMethods Integration Server.

<table>
<thead>
<tr>
<th>Purpose</th>
<th>File name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACL assignments</td>
<td>.\config\aclmap_sm.cnf</td>
</tr>
<tr>
<td></td>
<td>.\config\acllist.cnf</td>
</tr>
<tr>
<td></td>
<td>.\config\aclread.cnf</td>
</tr>
<tr>
<td></td>
<td>.\config\aclwrite.cnf</td>
</tr>
<tr>
<td></td>
<td>.\config\acls.cnf</td>
</tr>
<tr>
<td>Configuration information used for publishing and processing documents</td>
<td>.\config\dispatch.cnf</td>
</tr>
<tr>
<td>Event types and event subscriptions</td>
<td>.\config\eventcfg.bin</td>
</tr>
<tr>
<td>List of server errors and services that the webMethods Manager is to monitor</td>
<td>.\config\ISMJournalLogFilter.cnf</td>
</tr>
<tr>
<td></td>
<td>.\config\ISMSvcExcptFilter.cnf</td>
</tr>
<tr>
<td>Invoke path for the Integration Server</td>
<td>.\config\invokemanager.cnf</td>
</tr>
<tr>
<td>Port and associated protocols</td>
<td>.\config\port.cn</td>
</tr>
<tr>
<td></td>
<td>.\packages\WmRoot\config\listeners.cnf</td>
</tr>
<tr>
<td>Behavior when redirecting web automation</td>
<td>.\config\redir.cn</td>
</tr>
<tr>
<td>Partner server signon information</td>
<td>.\config\remote.cn</td>
</tr>
</tbody>
</table>
If you are also copying user packages, you should copy .\config\eventcfg.bin, which contains information for event types and subscriptions. Some of the services in the packages you copy might subscribe to events and therefore require this information.

In addition, you should copy all .access files that have been changed to the destination server (following the instructions in “Setting Up .access Files” on page 55).
Other Security Issues

In addition to the measures covered in this document, your security should include the following considerations:

- If your webMethods Integration Servers or related systems are co-located (for example, operated by an outsourcing company), you should consider their security architecture. For example, do they use switches to create virtual LANs (VLANs)? If so, how is traffic protected to prevent information from flowing between VLANs? How are their firewalls configured? Do they provide adequate protection for your servers? In general, the co-location company will need to provide greater security than you would provide yourself, because they are likely to also be hosting servers operated by your competitors, and hence are more vulnerable to insider attacks.

- Physical security of your webMethods Integration Server is critical. Apply the same protections to your server as any other mission-critical system in your environment to avoid theft, tampering, destruction, or water or electrical damage.

- Personnel security for your organization is critical. Because insider attacks make up a majority of security violations, consider how you verify and monitor employees. In particular, users with administrative rights (including anyone who is a member of the Administrators or Developers group) can do significant damage. Apply the same degree of personnel security as you do for administrators of other mission-critical systems.

- Denial of Service (DoS) attacks are virtually impossible to prevent. However, some firewalls can protect against certain types of attacks, such as those that open multiple connections to the webMethods Integration Server. Check with your firewall vendor to determine the types of DoS attacks that it can stop.

- As with any other software system, periodic maintenance is critical for a webMethods system. Be sure to download and install security related fixes and service packs promptly. Remove user accounts when individuals leave the organization or no longer have responsibilities for the webMethods products. Failure to remove unused packages, or to keep up with vendor-supplied patches, could ultimately prove to be a potential source of compromise.
Building Secure Applications

The methods described in this document can help make your webMethods Integration Server more secure. However, much of the security of your server depends on how carefully your build your application. This section provides a checklist of items to consider when designing, developing, testing, and deploying your webMethods Integration Server application. It is not complete, but rather a starting point for consideration.

- **Do services provide the minimum capability possible?** For example, rather than providing a general service to query a database (and relying on the calling user to invoke it only in authorized ways), develop a more specific service that will allow only an insert or query operation. This gives more control, and reduces the possibility of a user using the service in an unintended manner.

- **Does the service check all inputs for validity?** Do not rely on a calling user to validate the input. For example, if you are passing input to a shell or a SQL query, verify that all possible special characters have been removed, so that a malicious caller cannot cause unexpected results. In general, it’s better to allow known valid input, rather than trying to remove possible harmful input. That is, rather than trying to identify every character that could cause the underlying service to malfunction, if you know that a query should only contain letters, numbers, and spaces, allow only those characters instead of trying to remove all of the other characters.

- **Do you use “magic tokens,” and if so are they easy to guess?** Many systems create a magic token, such as a cookie, and use that as authorization for a future action. If the token isn’t truly random, then a malicious caller could guess the token, and thus gain access to another’s authorization. Note that tokens based on the time of day are common, but are not sufficiently random, because an attacker can guess the time very accurately.

- **Do you execute external programs?** Executing an external program on behalf of a caller is potentially a dangerous operation, because it may allow the caller to force your host system to execute software that may have unintended results. Avoid executing external programs if possible. If you cannot avoid it, be sure to carefully validate all input against what the program is expecting, so a caller cannot cause unexpected results. In particular, avoid using a shell as an intermediary, because shells perform significant parameter interpretation.

- **Do you access web sites on behalf of a caller?** If so, the service can be used as a way to attack another site. Be sure that if you access another site, you validate the input.

- **Do you handle input beyond the range of expected values?** Will your service behave in an unexpected fashion if input is larger than you intended? For example, what will the service do if the name or identifier of an item on a purchase order is longer than expected? Rejecting the purchase order may be acceptable; overflowing a buffer must not occur because it is likely to lead to opportunities for bypassing the webMethods Integration Server’s security.
Do you have any “debugging” capabilities that bypass normal controls? While debugging features are frequently helpful in the development process, they are frequently left in when a system enters production. Because such debugging capabilities frequently include back doors, it’s best not to put them in at all, or to ensure that they are protected as any other service is.

Are you relying on encryption to solve all your security problems? Encryption is a useful security feature, but it doesn’t solve all security problems. Verify that your application is not relying on encryption to cover up weak solutions to security problems.

Are you using good encryption capabilities? Almost anyone can come up with an encryption algorithm, but almost no one can come up with a good encryption algorithm. If you are using encryption for privacy, signatures, or any other purpose, use standard algorithms in standard ways. Proprietary encryption algorithms tend to not be as robust as standard ones.

Are you using different digital certificates for signing documents than for establishing SSL connections? It is preferable to have two different pairs of digital certificates and corresponding private keys: one pair for establishing SSL connections and a second pair for signing documents.

Do you rely on obscurity for your system security? If a system is truly secure, an attacker could have complete source code and still be unable to get in. Do not rely on an attacker being unaware of your design or implementation for protection.

Security Considerations for Packages

Keep the following security considerations in mind when you create and replicate packages:

- **ACLs.** Access Control Lists control who can access a package and its contents. The default settings might not be as restrictive as you need. See “Access to Resources” below for more information.

- **Replication.** You can associate a port number with a package so that when you replicate the package, the target Integration Server will make sure the port number exists on the target server. The port on the target server (whether it already exists there or is created dynamically) might be less secure than the one on the “originating” server.

**Access to Resources**

Access to a package and its contents is controlled through Access Control Lists (ACLs). An ACL consists of two group lists: a list of groups that are allowed access, and a second list of groups that are denied access. Each element can be associated with four ACLs: one List, one Read, one Write, and one Execute. For more information about ACLs, see “Understanding Users, Groups, ACLs, Folders, Services, and DSPs” on page 39.
By default, when a developer creates a package, the Integration Server assigns Default for the List ACL. (Packages do not have Read, Write, or Execute ACLs.) When a developer creates a top-level folder in the package, the Integration Server assigns Default for List, Read, and Write ACLs and Internal for the Execute ACL for the folder. Therefore, if a developer accepts the defaults, then the resultant package can be listed by any authenticated developer. In addition any authenticated developer can add, delete, lock and change ACL assignments for the elements in the package, and view service source and trigger conditions for Specifications, Schemas, and other elements.

Caution: When you add an element to an existing package, if you do not specify a particular ACL, the element inherits the ACL assignment from the parent folder. The parent folder's assignment might not be as restrictive as you want it to be.

Using the dictum “that which is not allowed is denied” is a good way to protect your system. Following this dictum, a developer should set the List ACL for a newly created package to Developer and set all four ACLs on a newly created folder to Developer. Later, the developer can modify the Execute ACL for the service so that a partner can execute it.

Further Reading

For further reading on building secure applications, see:


Auditing and Forensics

webMethods Integration Server can keep extensive audit logs that can be useful in determining what happened in case of attack. The webMethods Logging Guide describes the contents of logs and how to configure them. You can also refer to the “Detailed Auditing of Administrative Functions” Security Tech Note in webMethods Advantage.

Best Practice: Audit logs are only useful if you know they have not been lost or tampered with. To maximize their value, it’s best to store them off line by backing them up to tape or CD–R/RW media.

You may want to redirect your log files to a database or to an operating system log file (such as the syslog file on UNIX systems). You can use a custom event handler to redirect the webMethods Integration Server’s audit log. For details, see “Subscribing to Events” in the webMethods Developer User’s Guide.

Note: Be sure to back up log files regularly, and periodically move older log files off your system so that your file system does not fill up.

If an attacker were to gain access to your system, the integrity of your software and/or configuration files could be at risk. Many organizations are unable to detect or recover from such changes.

Best Practice: Before bringing an integration system into production, calculate cryptographic checksums of all software and configuration files that should not change. Do not include audit logs, password files, and other files that will change during normal operation. Keep the checksums off line (for example, on a floppy disk), along with a copy of the software used to calculate the checksums and compare their values.

There are both freeware and commercial products that can calculate and verify checksums. Tripwire (www.tripwire.com) is the best known product in this genre. Some security assessment tools such as Symantec’s Intruder Alert (www.symantec.com) also include similar capabilities. Finally, the md5sum command (available as part of the GNU utilities at www.gnu.org) provides similar capabilities. Do not use the sum command built into many systems, because attackers can easily modify programs or files without changing the checksums it generates.

Best Practice: Before bringing an integration system into production, make a full backup to tape or CD–R/RW media of all software and configuration files. Make a new backup every time the configuration changes. Doing so allows you to recover in case of a successful attack.
If a system is compromised, neither cryptographic checksums nor backups will help you know whether the system processed transactions incorrectly. If the attacker has not compromised the audit trails, they may help in tracking down the attack.

**Best Practice:** Don’t wait for an attack before you start creating and reviewing audit logs. Once the attack has occurred, it is too late to generate logs of what happened. Proactively generating and reviewing audit logs is the best way to prepare. Periodically verify that your webMethods Integration Server is generating logs, that they have sufficient information to determine what is occurring on the server, and that you know how to interpret them.

**Note:** webMethods has no knowledge that any of its customers’ systems have been attacked successfully. This recommendation is based on the premise that if such an attack were to occur, it would be too late to begin monitoring.

**Best Practice:** If you believe that a system has been compromised, immediately disconnect the system from the network to avoid further damage. Before examining the system to determine what happened, it is best to contact law enforcement authorities so that they can establish the compromised state before any corrective actions occur. A computer security forensics team can make effective backups that can be used for legal proof. Because any activity on the system can destroy evidence, dealing with a compromised system is best left to trained professionals.
Digital Signatures

Digital Signatures are useful for proving who sent a document and whether or not the document has been tampered with, either in transit or after it was received. A digital signature is a one–way hash of a document that was encrypted using the private key of the sender. The Integration Server supports S/MIME and PKCS#7 digital signatures, which are described in detail in the webMethods Built-In Services Reference Guide.

Although useful in securing your integration, digital signatures have some limitations:

- The digital signature only proves that the document was signed by software that has a copy of the private key used for the encryption. If the private key is stolen, it can be used to sign documents (create digital signatures) that are indistinguishable from digital signatures created by the authorized user of the private key. For this reason, it is critical to protect the private key. The Integration Server offers the option of storing the private key in a hardware device, which is then used for all signature operations. This option is more secure, because the private key cannot be retrieved from the hardware device, even if an attacker subverts the operating system.

  **Best Practice:** Be sure that the private key is stored in an operating system file that is not readable by users other than the user ID running the Integration Server.

  **Best Practice:** For highly secure environments, use the WmPKI package and hardware devices to perform digital signatures.

- Digital signatures are only as good as the data they sign. That is, if a digital signature only covers the total dollar amount on a purchase order, an attacker could substitute that dollar amount (with its signature) on a different purchase order for different items, with a different recipient, and a different date. For example, an attacker could steal the signed dollar amount field ($500,000) from one company’s purchase order for an airplane, copy the field to a purchase order for athletic shoes (originally $100) and illegally obtain $500,000 worth of athletic shoes. Thus, it is important that the signature cover all of the critical data. When designing an integration using digital signatures, consider the damage that could occur if modifications to any of the unsigned fields occur.

  **Best Practice:** Verify that the digital signature covers all critical fields in a document.

- Because digital signatures are intended to be used for long–term proof, it’s important to use a strong key. For normal business documents, a 1024 bit RSA key is adequate; for high value documents, or those where it is important to have proof that will still be valid for decades into the future, a 2048 bit RSA key is a good idea.
Legal proof for digital signatures varies from location to location, and from country to country. If you are relying on digital signatures to provide legal proof, consult an attorney knowledgeable in such matters. webMethods cannot tell you whether a particular digital signature method is adequate for your location.

Digital signatures provide proof only so long as the document that has been signed is unmodified. Because integration necessarily transforms documents (such as converting from XML or EDI to a database format), the digital signatures are necessarily broken. Thus, digital signatures are NOT an end-to-end solution.
FIPS Support

webMethods Integration Server Version 6.5 embeds the Entrust toolkit 7.0, which is FIPS 140–2 certified. FIPS (Federal Information Processing Standards) provides standards for information processing for use within the Federal government.

Note: The Integration Server itself is not considered to be FIPS 140–2 certified.

Running the Integration Server in FIPS 140-2-compliant mode ensures that it only uses FIPS compliant algorithms in the FIPS compliant modes. You can enable FIPS mode by setting the following extended setting on the Integration Server:

`watt.security.fips.mode=true`

See the “Managing Server Security” chapter in the webMethods Integration Server Administrator’s Guide for more information about the FIPS 104–2 compliant mode.
Contacting webMethods

webMethods welcomes your feedback on this document, on our other documents, and on our software. Please submit your comments to our Technical Services center:

- In US and Canada call 1-888-222-8215.
- In Europe call +800-963-84-637 or +31 20 77 83 640.
- In Asia call +612-8913-1198 or +65-389-3222.
- Worldwide, send an e-mail to support@webMethods.com.

If you think you’ve found a security vulnerability in a webMethods product, webMethods provides a security “hotline.” Send a description of the problem to security@webMethods.com. We will acknowledge your submission within 24 hours. This e-mail is for security bugs and vulnerabilities only; please submit security questions through the regular support line (above).

To be notified automatically of security alerts in webMethods products, send an e-mail with the text “subscribe” to security-alerts-request@webMethods.com. webMethods will not use this list for anything except security alerts.

Third-party penetration testing is one way to find flaws and identify misconfiguration in products. Some webMethods customers have expressed interest in performing penetration tests of their system, including their webMethods Integration Server. Penetration testing is not a replacement for careful configuration, or for careful design of your webMethods Integration Server application, but it can be a useful supplement to webMethods’ own security analysis and testing efforts. webMethods can suggest vendors that have experience with penetration testing of the webMethods Integration Server, and can assist you in the testing process. If you are interested in assistance with penetration testing, please send a summary of your interest to security@webMethods.com, and we will respond.

webMethods welcomes your feedback on additional security capabilities that would help your organization. Please send your suggestions to security@webMethods.com.

Appendix A. Externally Visible Services

This appendix shows how to use the Integration Server Administrator to easily create a list of services to be externally visible through a customized administrator, developer, or replicator port. Before performing this procedure, you must first create the customized port as described in “Customized Administrator–Only Port” on page 45, “Customized Developer–Only Port” on page 47, or “Customized Replicator–Only Port” on page 52.

**Note:** To determine the services that must be externally visible on the server for an adapter’s use, refer to the chapter for that adapter in the *webMethods Adapters Security Best Practices*. After obtaining the list, use the Edit Access Mode screen of the Integration Server Administrator to add the services to the allowed list to the appropriate ports. For additional information, refer to “Controlling Access to Resources by Port” in the “Managing Server Security” chapter of the *webMethods Integration Server Administrator’s Guide*.

**Important!** Do not log into the server through the port you want to change. The procedure involves temporarily denying access to all services through the port. If you log in on the port you want to change and then deny access to all services through it, you will be locked out of the server. Instead, log in through a different existing port or create a new port to log in on.

1. Open the Integration Server Administrator if it is not already open.
2. In the *Security* menu in the navigation area, click *Ports*.
3. Click *Edit* in the *Access Mode* field with which you want to work.
4. Click *Set Access Mode to Deny by Default*.
5. Click *Add Folders and Services to Allow List*.
6. Build a list of folders and services for the server to allow from this port.
   Use the pull down menu on the right of the screen to select the Administrators, Developers, or Replicators Execute ACL. The server displays a list of the folders and services protected by this ACL. Initially, all these items are selected. If you do not want to add all of them to the list, deselect the ones you do not want. (Use Ctrl–Click to deselect a selected item.) To move these entries to the list of folders and services that will be accessible through the port, click *Append Selected*. The server appends the selected entries the existing list. Then click *Save Additions*.
7. Click *Done* to return to the previous screen.

**Important!** The changes you make take effect immediately, even before you click *Done*. The function of the *Done* button is only to display the previous screen.
For additional information, refer to “Controlling Access to Resources by Port” in the “Managing Server Security” chapter of the *webMethods Integration Server Administrator’s Guide*. 
Appendix B. Special Considerations for Trading Networks

This appendix describes a number of security considerations to keep in mind if you run Trading Networks.

Additional Groups and ACLs

As part of installation, Trading Networks creates new Integration Server groups and ACLs. The new entities are:

- Group TNPartners with no initial members (users created for trading partners are put into this group)
- Group TNAdministrators with no initial members
- ACL TNPartners configured to allow members of TNPartners TNAdministrators and Administrators
- ACL TNAdministrators configured to allow members of TNAdministrators and Administrators
- By default, this allows administrators of the Integration Server unlimited access to Trading Networks functionality and data. You might want to consider refactoring the ACLs to separate Administrators and Trading Networks administrators.

Other security restrictions involving the TNAdministrators ACL are detailed in the “Document Security” on page 86.

Caution: Do not rename the TNAdministrators group. Changing the name will prevent administrative functions from operating correctly.

If you are upgrading from version 4.5.1 or earlier to version 6.5, you will notice that the TNGuest user and the TNGuests group and ACL no longer exist. These items were required for Profile Exchange, which was a feature that was removed from the Trading Networks in version 4.6.

Web Manager

Trading Networks ships with a web–based interface implemented with DSPs and services in the WmTNWeb package. This user interface was designed to be customizable if you need to provide flexible web access to your trading network.

Best Practice: If you are not using this web–based user interface, disable and (if possible) completely remove the WmTNWeb package from your Integration Server. If you have customized the user interface, follow the Integration Server security best practices to secure your new services and DSP pages.
List, Read, Write, and Execute ACLs

webMethods Integration Server has an ACL system that allows you to apply separate ACLs to a folder or to a service for each of the List, Read, Write, and Execute actions. The List, Read, and Write actions for all Trading Networks services are protected by the TNAdministrators ACL. The factory setting of the TNAdministrators ACL allows the Administrators and TNAdministrators groups. Only members of these groups will be able to view the list of Trading Networks folders and services, or be able to read or write to them.

Best Practice: If you want to give certain users the ability to list and read Trading Networks services, create a new group and ACL for this purpose (such as “TNDevelopers”). You can then use this ACL to grant List and Read access.

Caution: webMethods recommends that you do not grant Write access to Trading Networks services. Users with such access might maliciously or inadvertently change the behavior, or render unusable, your Trading Networks system.

webMethods also recommends that you do not grant Write access to Trading Networks folders. This will allow users to create new services in Trading Networks folders. These services can be lost when you upgrade to a newer version. Custom services should be created in a separate folder, not in the WmTN, WmTNWeb or WmTNSamples packages.

For more information about ACLs in the Integration Server, see the webMethods Integration Server Administrator’s Guide.

Profile Management

The Trading Networks Console uses services in the wm.tn.profile folder to manage trading partner profiles. The Execute action for all services in this folder are protected with the TNAdministrators ACL.

Best Practice: When using the Console to manage trading partners, always connect to the Integration Server using HTTPS. Confidential data about your business partners is transmitted between the Console and Integration Server, and this data should be transmitted only over a secure connection.

When the Console is used to create a trading partner profile, Trading Networks creates an Integration Server user account for the partner. The Console displays the account name and password after you create the profile. By default, the user account that Trading Networks creates for each trading partner uses the partner’s D-U-N-S® number as the user name for the user account. Trading Networks uses the partner’s D-U-N-S® number as the user name because, by default, the required ID type is set to “DUNS”.

Best Practice: If you want to give certain users the ability to list and read Trading Networks services, create a new group and ACL for this purpose (such as “TNDevelopers”). You can then use this ACL to grant List and Read access.

Caution: webMethods recommends that you do not grant Write access to Trading Networks services. Users with such access might maliciously or inadvertently change the behavior, or render unusable, your Trading Networks system.

webMethods also recommends that you do not grant Write access to Trading Networks folders. This will allow users to create new services in Trading Networks folders. These services can be lost when you upgrade to a newer version. Custom services should be created in a separate folder, not in the WmTN, WmTNWeb or WmTNSamples packages.

For more information about ACLs in the Integration Server, see the webMethods Integration Server Administrator’s Guide.
Document Security

During document recognition, Trading Networks extracts information from business documents to guide processing. In many cases, this information includes the external ID of the partner sending the document. As an additional security precaution, when documents are submitted from external clients (e.g., HTML clients or back-end applications), the sender’s external ID that Trading Networks extracted from the document is compared with the user logged in to the server.

The rules for this comparison are as follows:

1. If the document was submitted by a user with TNAdministrator privileges, Trading Networks performs no further checks. This allows an administrator to submit any kind of document.

2. If the sender of the document is unknown, or there was an error extracting the sender external ID from the document, no further checks apply (the sending partner will be the “Unknown” partner). This allows user-defined error handlers to process documents from unknown partners.
For all other documents, the external ID of the sending partner must be associated with the user who logged in to the server. This prevents malicious partners from pretending to be one another, or unknown third parties from sending documents claiming to represent a legitimate trading partner.

If a document fails this security check, no processing is performed. An error message is logged, the document is set into an ABORTED status, and an error is returned to the posting client.

**Best Practice:** Do not disable these security checks. If you develop your own version of `wm.tn:receive` (i.e. a top-level, publicly visible service to which your partners post their documents), consider using the `wm.tn.doc:checkUser` service to add this check into your own service.

**Caution:** These checks typically only apply to documents generated and submitted by external clients. If you apply this check to all documents processed through Trading Networks, you may encounter unexpected security errors under normal conditions. Consider the following process:

1. Your trading partner (Corp1) logs in to your server and submits an order (sender is Corp1). The document security check succeeds.

2. Your system then internally generates and submits an order response (sender is You.)

3. Because this action is being performed in the Corp1 login session, the document security check would fail and processing of the order response would be aborted.

Note that flat file TN document types can be configured to determine the document sender from the connected user. In this case, rather than extracting the sender’s external ID from the document, Trading Networks uses the name of the connected user. If the connected user is a valid trading partner, the user check described above is guaranteed to succeed.

**Security Certificates**

Trading Networks lets you specify security certificates that you can use for signing, verifying, encrypting and decrypting documents that you exchanged with your trading partners. This feature provides a very high level of document security.

The certificates and private keys used by Trading Networks are stored in a binary–encoded format in the Trading Networks database. The certificates and private keys used for signing outbound documents are also stored on the server’s file system for backward compatibility with earlier releases of Trading Networks.
When Trading Networks running one Integration Server polls Trading Networks running on another Integration Server, it connects and issues a query for all waiting documents, and then retrieves them one at a time for processing. This feature was implemented with an emphasis on security, but you should understand the issues involved. The exact scenario depends on whether your server is doing the polling or is being polled.

Polling

When Trading Networks running one Integration Server polls Trading Networks running on another Integration Server, it connects and issues a query for all waiting documents, and then retrieves them one at a time for processing. This feature was implemented with an emphasis on security, but you should understand the issues involved. The exact scenario depends on whether your server is doing the polling or is being polled.

When Your Server Polls a Partner's Server

A regularly scheduled job on your server executes `wm.tn.polling:remoteCheck`. This service logs on to your partner's server and queries for all documents waiting for pickup. Each of these documents in turn is downloaded from the partner's server using `wm.tn.doc:view`, submitted for internal processing using `wm.tn.doc:recognize` and `wm.tn:submit`, and acknowledged with another call to your partner's server.

This scheduled job that executes `wm.tn.polling:remoteCheck` runs with Administrative privileges. It logs on to your partner's server with the user name and password established in that partner's profile on your server. Documents submitted for internal

---

**Best Practice:** You should limit access to the certificates and private keys in the Trading Networks database and the server's file system.

1. Restrict access to the machine hosting the Trading Networks database to only users that absolutely require access.

2. Protect access to the Trading Networks database with strong passwords.

3. Do not allow unnecessary connections to the Trading Networks database.

4. Access to the SecurityData table in the Trading Networks database should be limited to database administrators and the user name that Trading Networks uses to connect to the database. This user name is specified in the “tn.store.db.user” property.

5. Access to the server's file system should always be limited as much as possible. This is especially true if you are using certificates to sign documents sent to your trading partners. The signing certificate files are stored in the `webMethods6\IntegrationServer\config\TNCERTSTORE` directory. Access to this directory on the server's file system should be limited to the server administrator.

6. The `pub.file:getFile` service is protected by the Internal ACL. Do not relax the ACL setting on this service if you are using certificates to sign documents sent to your trading partners.
processing are not processed with Administrative privileges; instead, before processing the document, the scheduled task assumes the user ID that corresponds to the sender of the document (Default if the sender is not recognized). This allows the document security check (described in “Document Security” on page 86) to work properly on documents that have been polled.

**Best Practice:** The default configuration for polling a partner’s server maintains the high level of security provided for externally submitted documents. If you want to maintain a very high level of security, you can take the following steps to secure the polling services from internal attacks:

1. Create a new user to perform polling (e.g., “TNPolling”). Set the Trading Networks property “tn.polling.user” to indicate that this user should be used for scheduled polling jobs (default is “Administrator”). Create a new group and ACL such that this user is the only one allowed.

2. Change the Execute ACL on the `wm.tn.polling:remoteCheck` service to use this new ACL and the settings on this service to enforce ACL checks at all times.

**When Your Server Is Being Polled**

At regular intervals your partner(s) logs on to your server and invoke `wm.tn.polling:localCheck`. This call returns a list of document IDs for all documents waiting for pickup by that particular partner. After this call, the partner will download each document, in turn, using `wm.tn.doc:view`. After processing each document on its own server, the partner acknowledges its successful receipt back to your server with a call to `wm.tn.polling:acceptDocument`.

To perform all these functions, your partner logs on to your server with the user name and password that you have established with that partner. The call to localCheck uses this login information to formulate a query that will prevent one partner from receiving documents meant for another partner.

**Best Practice:** You should restrict polling activity to a particular port on your server, and only allow the three services mentioned to be externally invoked on that port. For more information, see “Protecting Internal Services from External Access (Port Controls)” on page 53. Make sure the services involved have at least the TNPartners ACL on them.
Best Practice: The default configuration for being polled provides a high level of security by preventing malicious partners from retrieving each other’s documents. If you want to maintain a very high level of security, you can take the following steps to secure the polling services from internal attacks:

1. Create a new group (e.g. TNPolling). Only partners who are known to be polling your server should be members of this group. Create an ACL which allows only members of this group.

2. Change the ACL on the wm.tn.polling:localCheck and wm.tn.polling:acceptDocument to use this new ACL and the settings on this service to enforce ACL checks at all times.

Reliable Tasks

A reliable task is created when a service is invoked through a processing rule that specifies the reliable execution option, or when a document is scheduled for reliable delivery using a registered delivery service. A task manager handles scheduling these tasks, and deals with retries and errors. As soon as a task is ready to run and the resources are available to run it, it is executed.

By default these tasks are run with Administrative privileges. This allows services executing from a processing rule or a registered delivery service to run with Administrator privileges.

Best Practice: Change the Trading Networks property “tn.transport.user” to use another, less privileged user for executing reliable tasks. Alternatively, consider creating and using a new user specifically for this purpose (e.g. “TNReliableTask”)

Best Practice: To maintain a very high level of security, you should not only create a new user for this purpose, you should make sure that reliable services can only be invoked by this user and no other. Create a new group and ACL such that only this new user is allowed, and assign it to all reliably executed services (delivery services as well as those invoked from processing rules). These include the Trading Networks-supplied transport services (wm.tn.transport*). Additionally, make sure these services enforce ACL checks at all times.
**Best Practice:** When you initially set up a database to be used with Trading Networks, do not use a privileged account (such as the DBA or database administrator). Instead, use an account that only has the minimum level of access.
Appendix C. Common Criteria

The Common Criteria (CC) is an international standard (ISO 15408) that defines security requirements for software products. Products can be evaluated for conformance to the CC. Eight countries (Australia, Canada, France, Germany, Japan, New Zealand, United Kingdom, and United States of America) have testing laboratories that evaluate products for conformance with Common Criteria. In the U.S., the National Institute of Standards and Technology (NIST) established the National Information Assurance Partnership (NIAP) to encourage Common Criteria evaluation and accredit independent testing labs to validate claims of conformance to Common Criteria. webMethods selected Cygnacom, an Entrust company, as our testing lab. Common Criteria evaluations performed in any participating country are mutually recognized across 22 countries, including Austria, Australia, Canada, Czech Republic, Finland, France, Germany, Greece, India, Israel, Italy, Japan, The Netherlands, New Zealand, Norway, Singapore, Spain, Sweden, Turkey, United Kingdom, and United States of America. For more information about CC, see www.commoncriteriaportal.org/. An introduction to CC can be found at http://niap.nist.gov/cc-scheme/cc_docs/cc_introduction-v2.pdf

webMethods Fabric 6.5 has been evaluated using the Common Criteria. Information about the evaluated product information is listed on both www.commoncriteriaportal.org/ and niap.nist.gov/cc-scheme/vpl/vpl_type.html.

Most Common Criteria evaluations do not include all options and features of the product. The portion of the product that is included in the evaluated configuration is known as the Target of Evaluation (TOE). Customers who want to use webMethods Fabric 6.5 in the form in which it was evaluated must follow all the instructions in this appendix for configuring the TOE.

The information in this appendix is a supplement to both the information in the body of this guide, as well as to other webMethods documentation.

To understand the security claims made in the Common Criteria evaluation and how they are realized, read the webMethods Fabric 6.5 Security Target, the webMethods Fabric 6.5 Evaluators Report, and webMethods Fabric 6.5 Validators Report, all of which are available on Advantage under Best Practices > Product Security Information > Common Criteria.

Scope of the Evaluation

The TOE consists of the following portions of webMethods Fabric 6.5:

- Integration Server 6.5 (including Broker Administrator 6.5)
- Broker 6.5 with Service Pack 1
- Developer 6.5
- JDBC adapter 6.0.3 with Feature Pack 1
- JMS adapter 6.1

Use of any other webMethods components, including layered products such as Trading Networks, is not permitted in the evaluated configuration.

As with any Common Criteria evaluation, the webMethods Fabric 6.5 evaluation covers only a particular version of the software. All service packs and fixes identified in the evaluated configuration must be installed. Do not use service packs and/or fixes other than those explicitly listed in the evaluated configuration. The README for a fix will state whether or not the fix is part of the evaluated configuration.

In Figure 11, shaded areas represent the TOE for webMethods Fabric 6.5; unshaded areas represent portions of the environment that are not part of the evaluation.

![Figure 11. Target of Evaluation](image)

The Common Criteria evaluation covers the webMethods software; it does not include the hardware, operating system, or JVM, which are considered to be part of the environment.
In addition, the evaluation does not include drivers used by the JDBC adapter, or the
database system(s) to which the JDBC adapter is connected. The evaluation does not
include the drivers used by the JMS adapter, or any system with which the JMS adapter is
communicating.

In addition, the TOE does not include the following Integration Server features:

- Connectivity to NIS (Network Information Services) or LDAP (Lightweight Directory
  Access Protocol) servers for user authentication.
- Pluggable authentication, including connectivity to Netegrity SiteMinder.

The strength of the digital signature mechanisms has not been considered as part of the
evaluation.

In some cases, webMethods documentation refers to configurations or network
architectures that are not included in the evaluated configuration. Unless explicitly listed
in this appendix, you should assume that the configuration is not included in the
evaluated configuration. For example, descriptions of third party proxy servers and load
balancers from vendors such as Cisco, F5, and Juniper may be useful, but have not been
tested as part of the evaluated configuration.

Assumptions for the Evaluated Configuration

Every Common Criteria evaluation has a set of assumptions that define the environment
where the product will be used. The following assumptions were used in the evaluation of
webMethods Fabric 6.5:

- The administrator is trusted to correctly configure and operate the TOE according to
  the instructions provided by the TOE documentation.
- The TOE components critical to the security policy enforcement will be protected
  from unauthorized physical modification.
- There will be one or more administrators who are competent to manage the TOE and
  the security of the information it contains, and who can be trusted not to deliberately
  abuse their privileges to undermine security.
- Administrators will properly follow the instructions in this document and other
  webMethods documentation.
- Administrators will choose good passwords, and will not improperly disclose those
  passwords.
- There will be no untrusted software on the webMethods Integration Server and
  Broker.
- Users will protect their authentication data (that is, their passwords or private keys).
The TOE relies upon the IT environment to support protected communications, provide audit file protection, support partial domain separation, support non-bypassability, provide reliable time-stamps, and to perform user authentication when the IT environment is configured to do so.

Installation

Installation is described in the webMethods Installation Guide Version 6.5, webMethods JMS Adapter Installation Guide Version 6.1, and webMethods JDBC Adapter Installation Guide Version 6.0.3. Following is additional information to be aware of when performing the installation.

When you start the webMethods Installer program, the first screen offers a choice of which product version to install. As shown in Figure 12, select 6.5 and higher.

Figure 12. Release Selection Menu
After you enter your username and password, a menu provides the list of installable products, as shown in Figure 13.

![webMethods Installer 6](image)

**Figure 13. Overall Product Selection**

The following sections identify the specific options to use for each component of the evaluated configuration.

**Integration Server**

1. Check the box for Integration Server.
2. Make sure the following boxes are checked:
   - Program Files
   - Documentation
   - Broker Administrator 6.5 (optional - only if this Integration Server will be used to administer one or more Brokers)
3. Make sure the following boxes are unchecked:
   - Process Runtime
   - Repository Server (not shown in the figure).

4. Make sure all other boxes are unchecked.

   **Note:** The evaluated configuration contains no service packs for Integration Server.

Figure 14 shows how your installer page for the Integration Server should look when the appropriate boxes are checked. The specific items in the installer list may differ from those shown in the figure below, depending on the specific products you have licensed.

**Important!** Do not check any boxes other than the ones shown here.

---

**Figure 14. Integration Server Installation Options**

5. After installation of the Integration Server completes, you must download and install the WmCCaudit package from Advantage under **Best Practices > Product Security Information > Common Criteria**. See the *webMethods Integration Server Administrator’s Guide* for instructions on how to install and activate packages.
Broker

1. Check the box for Broker.
2. Make sure the following boxes are checked:
   - Program Files
   - Documentation
   - Service Pack 1 (with subitems Broker Client Java API, Command Line Tools, JMS Client API, and Program Files)
   - Broker Client Java API
   - Command Line Tools
   - JMS Client API

**Note:** The specific items in the installer list may differ from those shown here.

Figure 15 shows how your installer page for Broker should look when the appropriate boxes are checked.

**Important!** Do not check any boxes other than the ones identified here
Figure 15. Broker Installation Options

**Developer**

1. Check the box for Developer.

2. Make sure the following boxes are checked:
   - Program Files
   - Documentation
   - Samples (optional)

3. Make sure all other boxes are left unchecked.

**Note:** The evaluated configuration contains no service packs for Developer.

Figure 16 shows how your installer page for Developer should look when the appropriate boxes are checked. The specific items in the installer list may differ from those shown here.
**Important!** Do not check any boxes other than the ones shown here

---

**JDBC Adapter and JMS Adapter**

The JDBC adapter and/or JMS adapter rely on the Integration Server. You can install them at the same time as the Integration Server or afterwards.

1. Check the box for Adapters.
2. Make sure the following boxes are checked:
   - For JDBC:
     - Documentation and Program Files
     - Feature Pack 1
   - For JMS:
     - Documentation and Program Files
3 Make sure all other boxes are left unchecked.

**Note:** The evaluated configuration contains no service packs for the JDBC adapter and no feature packs or service packs for the JMS Adapter.

When you check the JMS adapter, the Broker Java Client API under Broker and JMS 6.5 will also automatically be checked. This is an included part of the JMS adapter.

Figure 17 shows how your installer page for Adapters should look when the appropriate boxes are checked. The specific items in the installer list may differ from those shown here.

**Important!** Do not check any boxes other than the ones shown here.

![Installer Page](image-url)

Figure 17. JMS Adapter and JDBC Adapter Installation Options
Supplementary Information for Logging Guide

All references to HTTP in the webMethods Logging Guide should be understood to include HTTPS, and references to FTP should be understood to include FTPS.

Network Protocols and Architectures in the Evaluated Configuration

In the evaluated configuration, you can configure the Integration Server to use the following protocols:

- HTTP/S with server certificates
- HTTP/S with client and server certificates
- HTTP over a Virtual Private Network (VPN)
- FTP/S with server certificates
- FTP/S with client and server certificates
- FTP

In all cases except HTTP/S with client and server certificates, authentication is performed using username and password. Note, in particular, that FTP/S with client and server certificates requires username/password authentication in addition to the client certificate.

Authentication using an HTML form-based login page is not included in the evaluated configuration.

Do not use the SMTP and POP3 network protocols in the evaluated configuration.

Communication between the Integration Server and the Broker must be protected by a VPN, regardless of whether the SSL option is used. In addition, you must configure the network topology to prevent any non-administrative users from accessing the network communications between the Integration Server and Broker.
Figure 18, Figure 19, and Figure 20 show the tested configurations for the Common Criteria evaluation.

Figure 18. Basic TOE Configuration with Physical Separation

Figure 19. Basic TOE Configuration with Procedural Separation

Figure 20. Reverse Invoke TOE Configuration
Auditing in the Evaluated Configuration

To meet the Common Criteria auditing requirements, the Integration Server must write an audit record every time it:

- Reads information from the audit records
- Unsuccessfully attempts to read information from the audit records
- Modifies the audit configuration while the audit collection functions are running
- Performs an operation on a package, folder, service, flow service, specification, schema, document type, or trigger
- Rejects any tested secret
- Modifies the behavior of the TOE security functions
- Modifies the values of security attributes
- Modifies the default setting of permissive or restrictive rules
- Modifies the initial values of security attributes
- Modifies the values of TSF data
- Modifies the group of users that are part of a role
- Uses the management functions

By default, no auditing is enabled; therefore, to ensure that the above events are recorded, follow the instructions in the webMethods Logging Guide to enable service logging. In the evaluated configuration, service logging must use a file, not a database. To ensure that all information is recorded, use webMethods Administrator to set the watt.server.auditLog parameter to brief or verbose. When auditing is enabled, the Integration Server also keeps track of when the wm.server.query:getPartialLog service is started and stopped.

When you first enable auditing, the change to the audit status does not take effect until the Integration Server is restarted. Therefore, you must immediately reboot the Integration Server after enabling auditing, to ensure that all subsequent events are properly audited.

Similarly, if you decide do disable auditing (by setting the watt.server.auditLog parameter to none), auditing is not disabled until the next Integration Server reboot. Therefore, you must immediately reboot the Integration Server after disabling auditing, to ensure that no further events are audited.

For wm.server.query:getPartialLog, the inputs are:

- log - the name of the log file to read
- numlines - the number of lines to return from the file
- startline - the starting line number in the file
- descendchecked - determines whether it uses ascending or descending order

Valid and default values:
- log - no default, any file in the "logs" directory is valid
- numlines - default is 35; any positive number is allowed
- startline - default is 0 (which means the last line); any positive number is allowed
- descendchecked - value "Ascending" or "Descending"

Errors from the auditing service are:
- If log is not a valid file name within the "logs" directory, the parameters are echoed back, and no audit data is returned
- If numlines < 0, a Java exception is returned
- If startline < 0, the value is ignored and treated as zero
- If descendchecked is anything other than the expected value, it is ignored and treated as Descending

By default only the Administrator has execute access to the wm.server.query:getPartialLog service. No other ACL should be associated with this service.

Every night at midnight (local time), the Integration Server closes the existing audit files and creates new audit files, as described in the *webMethods 6.5 Logging Guide*. Old audit files are not deleted automatically, but rather should be periodically backed up and deleted using operating system facilities.

As with all physical files containing data related to or supporting security functionality (including audit logs), the Administrator should be certain the IT environment’s access control measures prevent an attacker from accessing or modifying security-related data through the IT environment, as well as the webMethods product.

**Caution:** Recording every service invocation (as required in the evaluated configuration) will increase disk usage and reduce performance.

**Operating System and JVM Requirements**

Both the Integration Server and Broker must be run as non-administrative users. That is, under UNIX systems, the Integration Server and Broker must not be run as the "root" user or any other user ID with equivalent privileges. Under Windows systems, the Integration Server and Broker must not be run as the "Administrator" user, or any other user ID with equivalent privileges.
The operating system must be configured to protect the password file (webMethods6\IntegrationServer\config\users.cnf) against access by any user other than the non-privileged user ID that runs the Integration Server.

The security of the operating system and the JVM are outside the scope of the evaluation. However, the evaluated configuration relies on the operating system and JVM to support its security functions. As such, installing security updates to the OS and JVM are generally recommended, although the specifics of which updates should be installed may depend on your site policies and whether you are using an evaluated operating system.

**Configuring the Integration Server**

The Integration Server must be configured to perform application-specific actions, such as parsing documents, transforming the data, and passing the resulting documents to adapters.

**Required Configuration Settings**

1. Review the following sections in this document and make sure the evaluated configuration follows the guidance presented there:
   - Setting Correct ACLs on Services
   - Protecting Internal Services from External Access (Port Controls)
   - Setting Up .access Files
   - Controlling by IP Address or Domain

2. Change all built-in passwords, including those for Administrator, Developer, and Replicator, as well as the Master Password, which is used for protecting outbound passwords

3. Make sure the following configuration settings are in effect:
   - The watt.security.fips.mode option is set to **true**. See “FIPS Support” on page 80 for further information.
   - The watt.server.displayDirectories option is set to false.

**Users, Groups, and Passwords**

When you configure users and groups in the Integration Server, make sure each individual person has his or her own user ID and a corresponding password or certificate/key. For reasons of accountability (that is, so their actions can be distinguished in the audit trail), users must not share IDs, and must not share their authentication information (that is, password or private key).

The Integration Server allows the administrator to change the minimum password strength by adjusting the minimum number of upper case, lower case, numeric, and
special characters. Do not change the default settings for these values in the evaluated configuration.

Administrators must use a username and password, not a certificate, when authenticating to the Integration Server.

**ACLs**

Do not include non-administrative users in the Administrators or Developers groups, or any other group found on the Administrator or Developer ACL.

Use of the Anonymous ACL is prohibited except on those built-in services and DSPs that have the Anonymous ACL by default.

Configure the Integration Server to allow access only to those administrative and custom services and DSPs intended for external access. Restrict use of the Default ACL to those services and DSPs that should be callable by any authenticated user. Use of "deny by default" ports is preferred to "allow by default" ports. Use of IP and/or domain filtering is strongly encouraged. Use of administrator-only and developer-only ports (with no access to those services on ports used by untrusted users) is preferred to having a shared port that is used for both trusted and untrusted users.

**Custom Flow and Java Services**

Only built-in services and Flow services built from built-in services are included in the evaluated configuration. Do not use custom-written Java services.

Custom developed Flow services should ensure that all input is validated. For example, before using the JDBC adapter to execute dynamic SQL statements against a database, the Flow service must validate that any user-provided input does not contain strings that might be used to perform SQL injection attacks. Similarly, if a Flow service executes an external program, the Flow service must validate that any user-provided input could not execute a command other than that intended, and that the command executed does not have undesirable effects (such as deleting files or reformatting the disk).

It is the responsibility of the administrative staff (including any user who is a member of the Administrators and/or Developers groups) to ensure that the above configuration requirements are met, and that all input is validated.

**Broker Configuration**

During the installation process, configure the Integration Server to use SSL for all communications with the Broker. See *webMethods Integration Server Administrator’s Guide* for configuration instructions. After you configure the SSL setup, you must reboot the Integration Server, because SSL settings do not take effect until the Integration Server is restarted.
**Configuring the Broker**

During the installation process, configure Broker to use SSL for all communications with the Integration Server. See *webMethods Broker Administrator’s Guide* for configuration instructions, including instructions on how to disable all non-SSL ports. After you configure the SSL setup, you must reboot the Broker, because port settings do not take effect until the Broker is restarted.