

File Integrity Monitoring

Setup Guide

Contents:

- How File Integrity Monitoring Works
- Setting Up and Running a File Integrity Scan
- Creating or Customizing a File Integrity Policy
 - Specify Targets
 - Configure the Targets
 - Use Patterns to Create Inclusions and Exclusions
- Managing FIM Policies
- Setting Up Baselines
 - Add a Baseline
 - View Baseline Reports
 - Using Multiple Baselines
- Managing Ongoing Monitoring
 - Re-Baseline File Integrity Policies
 - Use the File Integrity Monitoring API
- Addressing FIM Findings
 - View FIM Scan Findings
 - Act on FIM Issues, Findings, and Events
- Best Practices for File Integrity Monitoring
- Appendix: Interpreting File Access Permissions

How File Integrity Monitoring Works



File integrity monitoring is a feature of CloudPassage® Halo® that protects the integrity of system and application software on your Linux or Windows cloud servers. It regularly monitors your servers for unauthorized or malicious changes to important system binaries and configuration files. Implementing file integrity monitoring can help you to

- Detect unauthorized intrusions into any of your cloud servers.
- Comply with mandates and standards such as PCI DSS, HIPAA, SOX, CSA, and SANS.
- Detect and repair tampering with your servers' system or application code.

Halo accomplishes file integrity monitoring by first saving a baseline record of the "clean" state of your server

systems. It then periodically re-scans each server instance and compares the results to that baseline. Any differences detected are logged and reported to the appropriate administrators.

The elements that make up the baseline include (1) cryptographic checksums (signatures) and standard metadata for all files being monitored, and (2) standard metadata for files without content (such as directories and symlinks).

If you are creating or editing a file integrity policy, note that there is no Log checkbox for a target. The absence of a "Log" checkbox means that every finding also generates an event. If it's the first time the finding (or event) has occurred, then an issue is created and tracked until that finding (or event) no longer occurs.

An administrator can inspect the metadata or the file itself on the server involved to understand the nature of the change and, if warranted, escalate the issue to an incident-response team.

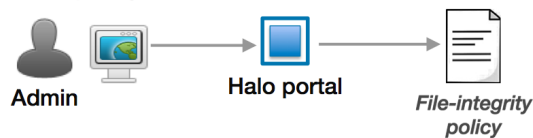
The first time Halo detects a specific file integrity anomaly, it creates an issue and a security event. If Halo then detects additional occurrences of that same finding, it updates the existing issue and event. Halo issues thus contain a record of all "bad file" findings of that specific type.

Halo file integrity monitoring involves these components and actions:

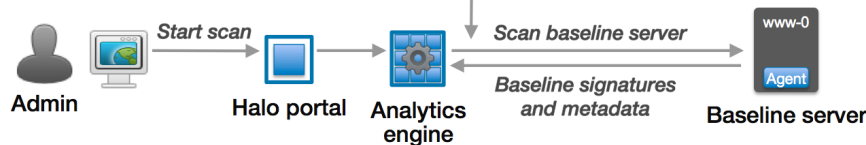
- **File integrity policy.** A security administrator uses the Halo portal to configure file integrity monitoring and to create a *file integrity policy* - essentially a list of paths to target objects (files and directories) to be monitored for changes.
- **Baseline server and baseline scan.** The administrator associates the policy with a *baseline server*—a server that represents the canonical, correctly configured, clean file structure of the cloud servers that will be scanned. Halo performs a *baseline scan* of this server, extracting and saving the cryptographic checksums (SHA-256 hash values) and metadata for all targeted objects on the baseline server. Halo then saves those *baseline signatures* and metadata for the policy.

Note: Halo allows you to define multiple baseline servers for a single policy, when the servers you need to scan are not all exactly identical.

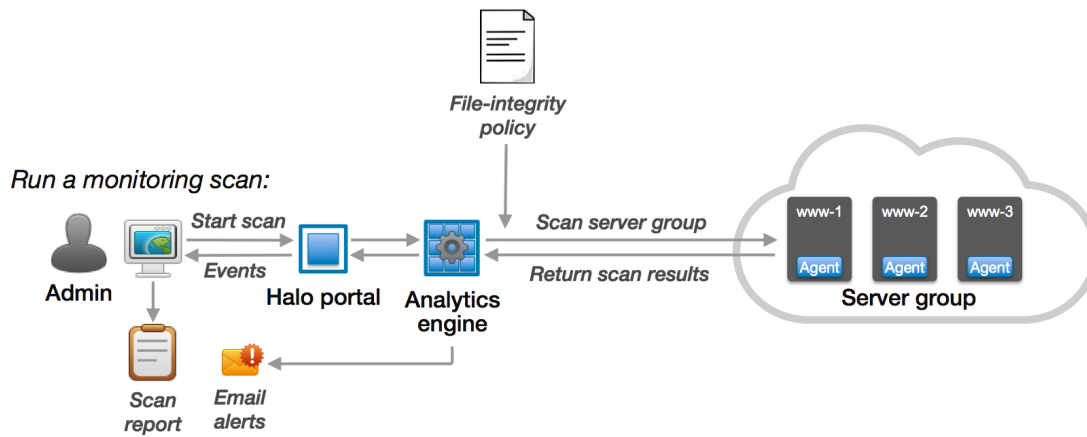
Create a policy:



Run a baseline scan:



- **Server group.** The administrator uses the Halo portal to assign the policy to a *server group*—an administrator-defined collection of servers that are identical to the baseline server in terms of system structure and configuration, at least for the targets specified in the policy.
- **Monitoring scans.** At a frequency determined by the administrator, Halo automatically runs *monitoring scans* of all servers in the group, including servers that come online automatically through cloning or auto-scaling. The Halo agent running on each server collects metadata and computes hashes of each targeted object on the server and sends them to the Halo analytics engine, which compares them with the baseline information, and reports any differences found to the Halo portal. Modifications, deletions, or additions of files or directories—as well as changes to metadata—are all detected.



- Security events and alerts. Halo records information on any detected changes as scan results, and also as *security events*. Administrators can view and act on those results and events in the Halo portal. Administrators may also receive email alerts triggered by designated high-priority events.

Setting Up and Running a File Integrity Scan

Getting your implementation of File Integrity Monitoring up and running involves creating a file integrity policy, setting up and scanning a baseline server or servers, and assigning the policy to a server group in your cloud.

1. Define a server group to scan.

After you have installed Halo agents on your servers, organize your servers them into groups along functional and architectural lines.

For information, In the *Halo Operations Guide* see [Setting Up Server Groups](#).

2. Select or create a file integrity policy.

You can create a completely new FIM policy, use one of Halo's policy templates as the starting point for the new policy, or base the new policy on a copy of one of your previously created FIM policies.

For information, see [Creating or Customizing a File Integrity Policy](#).

3. Specify a baseline server and run a baseline scan.

Baseline scans define the known-good server state against which other scans will be compared.

For information, see [Setting Up Baselines](#).

4. Assign the policy to a server group.

For information, in the *Halo Operations Guide* see [Assign Policies to a Group](#).

5. Choose and configure an automatic or manual scan.

For information, in the *Halo Operations Guide* see [Working with Scans](#).

6. View and act on scan results.

- To view FIM scan findings, in the list of scan results, click a result's Status column. In the *Halo Operations Guide*, see [View Scan Findings](#).

The scan findings page opens. See [Addressing FIM Findings and Issues](#).

- To view FIM issues, start with the Issues view on the Environment screen, then view the Issue Details Sidebar. In the *Halo Operations Guide*, see [Viewing Issues](#).

To get details of the findings underlying the issue, click the Scans link on the sidebar. The scan findings page

opens. For information see [Addressing FIM Findings and Issues](#).

Creating or Customizing a File Integrity Policy

The most efficient way to create a new File Integrity Monitoring policy is to use an existing FIM policy or policy template as the starting point for the new policy. You then assign a unique name to the policy, customize it as required, and save the new policy.

To create a new file integrity policy from scratch:

1. Display the *File-Integrity Monitoring Policies* page. In the Halo portal click Policies, then choose Policies > File Integrity Monitoring Policies.
2. Choose the starting point for the new policy.

For information about all policy creation options, see [Managing File-Integrity Policies](#).

3. Type the Name and Description of the new policy, then click Add Target to start specifying the policy's targets. Enter paths to one or more target files or directories (or Registry keys on Windows) that should be monitored; see [Specify Targets](#), below.

File or Directory	Recurse	Prelink	Description	Active	Critical	Alert	Delete
/etc/password	<input type="checkbox"/>	<input type="checkbox"/>	passwords	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Add Pattern							
/etc/shadow	<input type="checkbox"/>	<input type="checkbox"/>		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Add Pattern							
usr/bin	<input checked="" type="checkbox"/>	<input type="checkbox"/>	important binaries	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/> Exclude g*							<input checked="" type="checkbox"/>
<input type="checkbox"/> Exclude *pdf							<input checked="" type="checkbox"/>
Add Pattern							
Add Target							

[Save](#) [Cancel](#)

- You can specify the following objects as targets: individual files, directories, symbolic links, devices and special files (such as named pipes), and—on Windows—Registry keys.
- If you specify a directory, you can make the scan recursive (objects at all levels within the target directory are scanned) or non-recursive (only objects at the uppermost level within the directory are scanned).
- Within a directory target, you can either *exclude* objects that match a specific pattern, or you can *include only* objects that match the pattern.
- Mark or unmark the rule as active (will be scanned), critical (will be flagged in scan results), and/or alert (will generate an email alert). Make sure your policy has at least one active target.

For a list of restrictions on allowable targets in a policy and allowable kinds of information to scan, see [Limitations on Targets and Scans](#).

4. For each target, specify values for its flags; see [Configure the Targets](#), below.
5. When you are finished, click Save. The policy appears on its own page, along with a caution that the policy will remain inactive until you perform a baseline scan.
6. Click Add Baseline and then Request Baseline to run a baseline scan immediately: see [Setting Up Baselines](#), below.

7. Click Return to Policy List to return to the File Integrity Policies list.

Note: If you do not perform a baseline scan at this time, you can do it later by returning to the File Integrity Policies list, clicking Actions in the row for this policy, and selecting Baseline.

Specify Targets

A file integrity policy includes a list of target objects to be monitored for changes. When creating or editing a policy, use the Add Target link or the Delete icon (✖) to add or remove targets. Optionally provide a description for each target.

The following are the kinds of target objects you can specify, and the kinds of changes to each that a scan can detect.

File type	Added/Deleted	Content	Metadata	Target path
Text or binary file	Yes	Yes	Yes	
Directory	Yes		Yes	
Symbolic link ¹	Yes		Yes	Yes
Device/special file	Yes		Yes	
Windows Registry key	Yes	Yes	Yes	
1. See note (below)				

Note: For devices and special files (such as named pipes), only additions, deletions, and metadata changes are detected.

Note: For symbolic links, changes to the target specification are also detected—although changes to the target file itself are not detected.

Here are some examples of targets:

- Individual binary or text files. For example:

Linux:

```
/vmlinux
/usr/sbin/httpd
/etc/passwd
```

Windows:

```
C:\Windows\regedit.exe
C:\Program Files\WinZip\WINZIP64.EXE
%SYSTEMDRIVE%\Program Files\WinZip\WINZIP64.EXE (system environment variables supported)
```

Important: In Windows, the names of all target files (except symbolic links) must include the file extension.

- Directories and their contained objects (targets are at top-level only if non-recursive, at all levels if recursive).

For example:

Linux:

```
/bin      (objects in the /bin directory)
/etc      (objects in / etc)
```

Windows:

```
C:\Windows\System32 (objects in the System subfolder of the Windows folder)
C:\Program Files\   (objects in the Program Files folder)
```

- Windows Registry keys (on Windows only):

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell
```

When Halo performs a baseline scan using the target expressions in the policy, it creates a checksum (signature) for each text or binary file and records it, along with the directory in which the file was found. For all scanned objects, Halo records values for the following metadata:

Linux:

- user owner
- group owner
- permissions
- ctime
- mtime

Windows:

- owner
- permissions (access control list)
- Alternate data streams content
- creation date-time
- modification date-time

Subsequent scans will then detect whether the content of any of the files has been altered, whether any object has been deleted from or added to any monitored directory, and whether any critical metadata (owners or permissions) has changed. Any of those changes are reported as findings in the scan results and as security events.

Limitations on targets and scans

- Halo cannot scan more than 20,000 objects per server, and it does not analyze individual files of 1 GB or larger. This also means that a file integrity baseline cannot contain more than 20,000 objects.
- A file integrity scan will not return more than 10,000 failed objects per scan per server. In the event of a catastrophic scan in which more than 10,000 scanned objects fail, only the first 10,000 will be returned.
- Halo will not scan a target that is the directory `/proc` or any of its contents.
- Using the CloudPassage API to create a file-integrity policy with more than a thousand defined targets might result in (1) a timeout failure of the API method that creates the policy, (2) a failure of the baseline scan to complete, or (3) severe slowdown of the portal UI when trying to display the policy.

For best performance, CloudPassage recommends that you use recursion (with exclusions and inclusions as needed) to scan up to thousands of objects, while keeping the number of defined targets (= lines in the policy) below a thousand.

- CloudPassage recommends that you do not scan files that change often, such as log files, active database data files, and email files.

Configure the Targets

Every target in a file integrity policy has several associated flags that control recursion and event-related features:

- **Recurse.** Select this checkbox to enable recursive scanning of a directory target. A recursive target includes all subdirectories at all levels within the target directory (unless you have defined exclusions for the target). A non-recursive target includes only the objects at the top level of the directory. (Default = non-recursive.)
- **Prelink.** *[Linux only]* If this rule's target may include binary files (specifically, shared libraries in ELF format), and if pre-linking may be enabled for those files, select this checkbox. If it is selected, Halo accounts for pre-linking whenever it scans a shared library within the scope of this target. Do not select the Prelink checkbox if this target contains no ELF libraries or if pre-linking is disabled.
- **Active.** Select this checkbox to include this target in future scans. Deselect it to temporarily disable this target, without actually removing it from the policy. (Default = active.)

Important: When creating or editing a file integrity policy, always leave at least one target active. You cannot save a policy that has no active targets.

- **Critical.** Select this checkbox to specify that changes to this target should be flagged as critical. (Default = non-critical.)

Critical events are flagged with a special icon on the scan results page and on the Security Events History page, and you can sort the list to make those events appear at the top.

- **Alert.** Select this checkbox to specify that, if any changes are detected in this target, an email notification should

be sent to the users specified in the alert profile(s) of the server group to which this policy is assigned. (Default = no alert sent.)

Hint: Because a scan group can have more than one alert profile, you can for example configure Halo to send critical events to a different set of administrators than non-critical events.

Use Patterns to Create Inclusions and Exclusions

If a target in your policy is a directory, by default Halo scans all monitored objects within that directory (and all of its subdirectories, if the **Recurse** checkbox is selected). You can refine the set of objects scanned for that target in two ways—either by specifying that only certain objects can be included in the scan, or by specifying that certain objects must be excluded from the scan.

Both inclusions and exclusions are defined as search-string patterns. If the pattern matches the name of any monitored object within the target, that object is scanned if the pattern is an inclusion, and not scanned if the pattern is an exclusion.

Patterns can have wildcards. Two wildcards are supported: ***** (representing zero or more of any characters) and **?** (representing exactly one of any character). Thus, for example, the pattern **logs** will match any file or directory named exactly **logs** (case-sensitively on Linux, case-insensitively on Windows); and the pattern ***.log** will match any file whose filename extension is **.log**.

Defining an Inclusion

To make sure that your scans include just a certain set of objects within your target, specify as the inclusion a search pattern that will match all objects in the set but no others. For example, if you are interested in scanning only executable files and code libraries within a particular target directory on Windows, you can specify ***.exe** and ***.dll** as two inclusions.

To add an inclusion to your file integrity policy, click the **Add Pattern** link for the target directory, then move the slider to read "Include". Enter a string or pattern representing the inclusion. You can add any number of inclusions to a target.

File, Folder, or Registry Key	Recurse	Description
C:\Windows\system32	<input checked="" type="checkbox"/>	32-bit system files
<input type="text" value="*.dll"/>	<input type="checkbox"/>	dynamic link libraries
<input type="text" value="*.exe"/>	<input type="checkbox"/>	executables

[Add Pattern](#)

Defining an Exclusion

To avoid scanning a certain set of objects within your target, specify as the exclusion a search pattern that will match all objects in the set but no others. For example, if you do not wish to scan PDF files within a particular target directory, specify ***.pdf** as the exclusion. (On Windows, that will match all files ending with **.pdf** or **.PDF**. On Linux, you might want to define another exclusion that is ***.PDF**.)

To add an exclusion to your file integrity policy, click the **Add Pattern** link for the target directory, then move the slider to read "Exclude". Enter a string or pattern representing the exclusion. You can add any number of exclusions to a target.

Targets

File or Directory

Recurse

Description

/bin

Exclude

*.java

Exclude

*.cron

Add Pattern

☒

System binaries

Exclude Java files

Exclude cron files

Mixing Exclusions and Inclusions in the Same Target

You might not often mix inclusions and exclusions in the same target, but it is possible to do so. To understand how they will interact, keep in mind that exclusions take precedence. For example:

- With multiple exclusions in a target, an object that matches any of them is excluded.
- With multiple inclusions in a target, an object that matches any of them is included.
- With both exclusions and inclusions in a target, an object that matches both an inclusion and an exclusion is excluded. For example, within a target directory, you can include the `bin` subdirectory but still exclude all files in that subdirectory with the extension `.txt`.

Note that the inverse is not possible; an inclusion within an exclusion is still excluded. For example, if the `bin` directory is excluded, and `.txt` files are included, any `.txt` files within `bin` will *not* be scanned.

Managing FIM Policies

As your security requirements grow and change, you will also want to use Halo's policy management options to manage the policies that are or are not included in the list of active policies.

This section describes Halo's policy actions. To perform most of the following actions, you choose an active policy, policy template, or retired policy from a list and click its Action button. Then you select an action from the drop-down list.

CloudPassage HALO

Environment

Events

Policies

File Integrity Policies

Active Policies | Policy Templates | Retired Policies

Add New Linux Policy

Add New Windows Policy

Import Policy

Search

Name ▲	OS Type	Description	Used by Group(s)	Active Baselines	Actions
Core Registry Keys (Windows 2012) v1 (Copy)	Windows	This policy has been designed to detect changes in Windows Server 2012 registry keys that might indicate malware or other evidence of system compromise. These settings can include network, user behavior, administration, audit, and several others.	R&D Northwest	0	<div>Actions</div> <div>View</div> <div>Edit</div> <div>Export</div> <div>Retire</div> <div>Clone</div> <div>Delete</div>
Core Registry Keys (Windows 2012) v2	Windows	Recommended for: General system security, PCI-DSS, HIPAA, SANS 20, SOC 2 This policy has been designed to detect changes in Windows Server 2012 registry keys that might indicate malware or other evidence of system compromise. These settings can include network, user behavior, administration, audit, and several others.	No group assigned	0	
Core System Files (CentOS Linux / CentOS)	Linux	Recommended for: General system security, PCI-DSS, HIPAA, SANS 20, SOC 2 This is a FIM policy for CentOS Linux systems and will work with CentOS 5, CentOS 6, and CentOS 7.		0	

Policy Manipulation Actions:

8

Export a Policy

To export a policy from Halo, select "Export" from the Actions dropdown list.

Halo saves the policy's settings as a JSON-formatted file. You can securely archive the policy file, share the policy with other Halo users, or re-import it at a later time.

Import a Policy

To import a policy into Halo, click the Import Policy button above the policy list.

On the Import page, click Choose File, then navigate to and select the desired JSON-formatted policy file to import. If the import is successful, the imported policy appears in your list of active policies.

Retire a Policy

To retire a policy, select "Retire" from the Actions dropdown list.

Retiring a policy removes it from Halo's list of active policies and adds it to the list of retired policies. Retired policies are available for later use if they are unretired.

Unretire a Policy

To unretire a policy, in the Retired Policies page select "Unretire" from the Actions dropdown list.

Halo moves the policy from the retired policies list to the active policies list. It is once again available for editing or assigning to server groups.

Delete a Policy

To delete a policy, select "Delete" from the Actions dropdown list, then in the confirmation dialog click OK.

Halo permanently removes the policy. It no longer appears in the Active Policies page and cannot be retrieved.

Note: The only way you can recover a deleted policy is to have exported it first, so that you can re-import the exported file.

Policy Creation and Editing Actions:

Clone a Policy or Template

To clone a policy or policy template, select "Clone" from the Actions dropdown list.

Halo creates a copy of the policy, adds the word *Copy* to the policy's name, and places it in the Active Policies list. You often can use the cloned template as-is, or you may wish to use it as the starting point for a custom policy. In that case, create a unique name and description for the new policy, then customize its rules.

Note that Halo will not permit you to save a cloned policy if it does not have a unique name.

Create a New Policy

To create a new policy, click the Add New Policy button above the policy list.

On the new Policy page, Create a unique name and description for the policy. Initially, the policy is empty; add rules as desired.

Halo will not permit you to save a new policy until you assign the policy a unique name.

Edit a Policy

To edit an active policy, select "Edit" from the Actions dropdown list.

The Edit Policy page opens, on which you can change the policy's name, description, and rules. When you save it, the updated policy appears on the Active policies page.

Setting Up Baselines



Before you can use a file integrity policy or assign it to a server group, you need to run a baseline scan on it. To do that, you need to assign a baseline server to the policy.

Also, you will need to re-run a baseline scan whenever you make changes to the baseline server's target objects, or the policy that the baseline server is assigned to.

A *baseline server* represents the golden master—the canonical, correctly configured, clean system of the server group that you will assign the policy to. The baseline server could be one of the servers in that server group, or it could be a server set up solely as a template for the correct configuration of that type of server.

The baseline scan is run only on the baseline server; subsequent monitoring scans are run on all the servers of the policy's server group. Therefore, the structure and content of all servers in the group should in general be identical to the baseline server—at least for the specific file targets defined in the policy. (Exceptions to this are possible; see [Using Multiple Baselines](#).)

Add a Baseline

Immediately after saving a new file integrity policy or saving changes to an existing one, you are prompted to request a baseline scan. You can do it then, or you can later navigate to the File Integrity Policies list and select that policy. Then do this:

1. In the Halo portal, click Policies > "File Integrity Policies", select a policy then in the policy's details page click Add Baseline.
2. In the Select Baseline Server dialog box, use the dropdown list to choose the server that you want as the baseline. The list includes all of your currently online servers that have an installed Halo agent and are of the same general operating system (Linux or Windows) as your policy.

The screenshot shows a dialog box titled "Select Baseline Server". It features a dropdown menu for "Baseline Server" with "devops-balancers1-3" selected. Below this is a section for "Baseline expires" with radio button options: "Never", "30 Days", "90 Days" (which is selected), "180 Days", and "Custom: [] days". At the bottom, there is a "Comment" text area containing the text "Valid until Q3 update" and a "Request Baseline" button.

Select the lifetime of the baseline scan (the number of days before it expires), and optionally add a comment about this baseline server.

3. Click Request Baseline to start the baseline scan.

Depending on the number and size of the targets in your policy, running a baseline scan can take several minutes

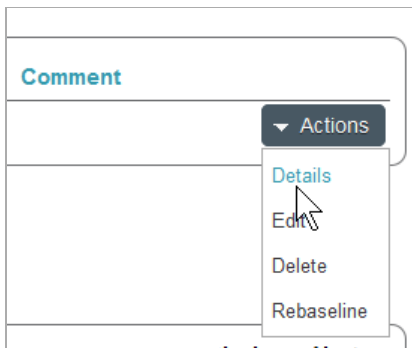
or longer.

After the baseline scan has completed and shows a status of *Active*, you can assign the policy to a server group and start running file integrity scans.

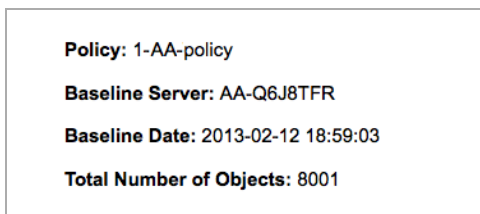
View Baseline Reports

Every time you run a baseline scan, Halo generates a report listing all of the target elements that were scanned on the baseline server. You can access that report at any time to verify that your file integrity policy correctly specifies all the targets that you want to scan and that your baseline server contains all of those targets.

1. To view a baseline report, click **Actions** for a given baseline server in the **Baseline** area of the **File Integrity Policy** page.



2. Select "Details" from the drop-down list. The **File Integrity Baseline Results** page appears, displaying at the top of the page the baseline's policy, the baseline server's name, the date of the baseline scan, and the total number of objects scanned for the baseline.



IMPORTANT: A baseline cannot contain more than 20,000 objects. Halo will invalidate a baseline scan that would return more than 20,000 objects.

For each top-level target in your policy, the baseline report displays:

- The target path, the inclusions and exclusions defined for the target, and the total number of objects scanned within the target.
- A line of information for each individual scanned element or sub-element of that target. The information includes the full path and type (directory, file, or link) of the element, plus its metadata and its cryptographic signature or linked-to target.

Target: C:\Program Files (x86)\Common Files\System
Inclusions: *.dll
Exclusions:
Number of Objects: 2

Object	Type	Owner	Access	Created / Modified	Content
C:\Program Files (x86)\Common Files\System\wab32.dll	file	NT SERVICE\TrustedInstaller	NT SERVICE\TrustedInstaller:(Allow)(RD,WD,AD,REA,WEA,X,DC,RA,WA,DE,RC,WDAC,WO,S) BUILTIN\Administrators:(Allow)(RD,REA,X,RA,RC,S) NT AUTHORITY\SYSTEM:(Allow)(RD,REA,X,RA,RC,S) BUILTIN\Users:(Allow)(RD,REA,X,RA,RC,S) APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:(Allow)(RD,REA,X,RA,RC,S)	2014-11-18 23:17:56 2014-10-29 01:09:52	062fe46a6a33 6ab087ee85d4 bcc7c45401cc e6b2fb18c3ec 03311adae761 6cc6
C:\Program Files (x86)\Common Files\System\wab32res.dll	file	NT SERVICE\TrustedInstaller	NT SERVICE\TrustedInstaller:(Allow)(RD,WD,AD,REA,WEA,X,DC,RA,WA,DE,RC,WDAC,WO,S) BUILTIN\Administrators:(Allow)(RD,REA,X,RA,RC,S) NT AUTHORITY\SYSTEM:(Allow)(RD,REA,X,RA,RC,S) BUILTIN\Users:(Allow)(RD,REA,X,RA,RC,S) APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:(Allow)(RD,REA,X,RA,RC,S)	2013-08-22 04:17:08 2013-08-22 04:17:00	31fa00a998b1 54f03e0c6bb6 ddfed638244 e99b14d673fd 709cd3377d2b 68

The report includes one table for each top-level directory target specified in the policy. For example, if the policy contains target paths starting with /bin, /etc, and /usr, the report will include three tables of scanned elements.

- Examine the pathnames and metadata in the report to satisfy yourself that the file integrity policy specifies the appropriate elements for ensuring the integrity of critical files, and that it does not waste time scanning unimportant elements. If necessary, refine the policy by modifying targets and inclusions/exclusions.

Using Multiple Baselines

In some situations a server group consists of servers that are similar but not in all cases identical. For example, patch levels or application versions might vary slightly among the servers. To help you handle that situation without fragmenting your server groups, Halo allows you to define several baseline servers for a single group. The baseline servers together must cover all acceptable configurations of the group's servers.

When you run a file integrity scan on a server group with multiple baselines, each target object's signature and metadata are compared with the signatures and metadata of that object on each of the baseline servers—and if a match occurs with any of them, the target rule is matched. Specifically:

- For a changed object, if none of the baselines matches the target object, it is considered a violation and a security event is triggered.
- For a deleted object, if all of the baselines contain the target object and the scanned server does not, it is considered a violation and a security event is triggered.
- For an added object, if none of the baselines contains the target object and the scanned server does, it is considered a violation and a security event is triggered.

Specifying additional baseline servers for a file integrity policy is as simple as specifying the first one. On the File Integrity Policy page, click **Add Baseline**, choose an expiration time, and select the server to be the baseline. After you click **Request Baseline** to perform the baseline scan, the new server appears in the policy's list of baseline servers.

Baselines				
Status	Baselined	Baseline Server	Expires	Comment
Active	2015-03-23 18:20:05	2012-R2-workgroup	Never	
Active	2015-03-23 18:12:35	Win-2012-R2-DC	Never	

If you make any changes to your file integrity policy, all baselines in effect at that time become invalid. You will need

to re-baseline all invalid baseline servers before you will be able to run a file integrity scan with that policy.

Managing Ongoing Monitoring



After you have run the baseline scan for a policy, assigned the policy to a server group, and then manually scanned your servers, you can view the results to address security events and alerts, and to manage updates to file integrity settings and policies.

Note: Going forward, be sure to set up automatic scanning to make sure that your servers are regularly examined for file integrity issues.

Re-Baseline File Integrity Policies

Whenever you alter a target in a file integrity policy, the policy's existing baselines involving that target are invalidated. You must re-run the baseline scan for any affected baseline server, or add a new baseline. Note that re-running a baseline is not required during the normal elastic operation of your cloud, because Halo automatically accounts for servers that come online or go offline due to server cloning or auto-scaling.

Whenever you make a configuration change, addition, or deletion to any of the monitored objects in a policy's server group, you must make the change to the appropriate baseline server itself, propagate that change to all the servers in the group, and then re-run the baseline scan for that policy.

To re-run a baseline scan, go to Policies > File Integrity Policies in the Halo portal and locate the policy that you wish to re-baseline.

Whenever you alter the targets in a policy, or whenever changes, additions, or deletions are made to the scanned files in that policy's server group, you must re-run a baseline scan for that policy.

To re-run a baseline scan:

1. Open the File Integrity Policy page for the policy that you want to re-baseline.
2. In the Baselines area, click the Action button in the row for the baseline that you want to re-run, and select Re-baseline.

Note: Whenever you re-baseline a policy, you have the opportunity to assign a different baseline server.

A success message is displayed, meaning that the policy's baseline server is being re-scanned. When the process is finished, a "File Integrity baseline" event is created and is visible on the Security Events History page.

Use the File Integrity Monitoring API

You can use the CloudPassage API to automate file integrity monitoring or build its capabilities into your own security tools. The API includes the following file integrity modules and functions:

- File Integrity Policies API. Enables you to list all policies, get the details of a single policy, create a policy, update a policy, and delete a policy.
- File Integrity Policy Baselines API. You can list all baselines for a policy, list a single baseline, create a baseline (run a baseline scan), delete a baseline, and request a re-baseline (re-run a baseline scan).
- Assign a file integrity policy. The Server Groups API enables you to assign one or more file integrity policies to a server group.
- File integrity events API. You can use the FIM API to retrieve File integrity events from the Halo database.

See the [Halo REST API Developer Guide](#) for details.

Addressing FIM Findings

To accurately assess the level of risk associated with a given FIM finding, issue, or event, you may need to examine the details of one or more FIM scan findings.

As described in [Setting Up and Running a File Integrity Scan](#), the easiest way to examine FIM findings in detail is to click a Scan's Status column to display a summary of the scan's results. From there, you can view the individual findings within a given result.

To accurately assess the level of risk associated with a given FIM event, you need to examine the event's details.

View FIM Scan Findings

The FIM scan results page lists the results of all the all of the "bad" findings detected in a file integrity scan.

Actions: [Server Scan History](#) | [Launch New Scan](#)

Target	Added	Modified	Missing	Ok	Result
▶ /dev	22	2	3	450	Failed
▼ /etc	2	23	13	159	Failed

▶ Added : /etc/centos-release

▶ Added : /etc/hostname

▼ Modified : /etc/fstab

	Date	Server	Owner	Group	Permissions	Content	Object ctime	Object mtime
Baseline object metadata	2014-05-20T22:14:53Z	ip-10-198-25-7	root	root	rw-rw-rw		2013-12-12 20:58:08	2013-12-12 19:26:16
	2013-07-08T20:34:17Z	ip-10-161-23-175	root	root	rw-rw-r--		2013-05-09 18:53:44	2013-05-09 18:53:29
Scanned object metadata	2014-05-21 17:16:38	04_JCTF-3	root	root	rw-r--r--		2013-03-11 10:22:48	2013-03-11 10:01:49

Last Baseline: 2013-07-08 20:34:17
Baseline Server: ip-10-161-23-175.us-west-1.compute.internal

▶ Modified : /etc/group

▶ Modified : /etc/group-

- Each row is a scan finding — the results from applying a single policy rule to the server. Findings are grouped by policy; all rules from all policies used in the scan are included.
- Issues are grouped by policy rule; all rules from all policies used in the scan are included.
- By default, critical issues appear first, then non-critical, then rule passes.
- Expand a rule to view a list of its individual issues (scanned objects). Pink objects are failures of the rule; green objects are passes.
- Within a rule, scanned objects are by default sorted in this order: added, modified, missing, OK.
- Expand an object to view how it compares to its baselines.

View FIM Finding Details

For a given finding or event, click [More details](#) to view both the baseline (original) metadata and current metadata

for the file.

- For Linux, Halo reports the following metadata items for each scanned file:
 - Owner. Username of the owning account.
 - Group. Name of the owning group
 - Permissions. See [Linux Permissions](#) for more information.
 - Content. Final 8 digits of the cryptographic signature of the file content (SHA-256 hash).
 - Object ctime. The last time the file was written to, or any properties or metadata were changed.
 - Object mtime. The last time the file was written to.

File Integrity object missing	about 1 hour ago	Directory /tmp/symtest is missing on Linux server ip-10-122-50-65 (184.72.64.178) (source: 18-Feb-linux)							Add Exception
	Date	Server	Owner	Group	Permissions	Content	Object ctime	Object mtime	
Baseline object metadata	2013-02-19 00:42:03	usha-qa-test3	root	root	rxwx-r-x		2013-02-19 00:35:39	2013-02-19 00:35:39	

Last Baseline: 2013-02-19 00:42:03
Baseline Server: [usha-qa-test3](#)

- For Windows, Halo reports the following metadata items for each scanned file:
 - Owner. User or group.
 - Permissions. See [Windows Access Control Entries](#) for more information.
 - ADS. Filename and content signature of any alternate data streams content attached to the target.
 - Content. Final 8 digits of the cryptographic signature of the file content (SHA-256 hash).
 - Created. When the file was created.
 - Modified. The last time the file was written to.

File Integrity object added	22 minutes ago	File C:\FEB-20\feb20test.txt was added on Windows server AMAZONA-Q6J8TFR (184.169.255.91) (source: 111-Windows-FIM)							Add Exception
	Date	Server	Owner	Permissions	ADS	Content	Created	Modified	
Scanned object metadata	2013-02-20 20:09:59	AMAZONA-Q6J8TFR	BUILTIN\Administrators	NT AUTHORITY\SYSTEM:(I)(Allow)(RD,WD,AD,REA,WEA,X,DC,RA,WA,DE,RC,WDAC,WO,S) BUILTIN\Administrators:(I)(Allow)(RD,WD,AD,REA,WEA,X,DC,RA,WA,DE,RC,WDAC,WO,S) BUILTIN\Users:(I)(Allow)(RD,REA,X,RA,RC,S)	feb20ads.txt 0e8c03ad	7852b855	2013-02-20 20:07:40	2013-02-20 20:07:48	

Last Baseline: 2013-02-20 20:00:00
Baseline Server: [AMAZONA-Q6J8TFR](#)

Changes to Object ctime and Object mtime on Linux, or Created and Modified on Windows, are not considered critical. If these values do not match the baseline but the other metadata and the signature still match, no event is created.

The details drop-down also includes the date of the last baseline scan and a link to the Server Summary page of the server for each of the active baselines of this server group.

A file integrity finding is reported whenever a policy target has changed through alteration of its content, ownership or permissions, or when a subdirectory or file has been added to or removed from a directory target. The above details show that the target file `sethc.exe` has changed in these ways:

- Its ownership has changed, from "NT SERVICE\TrustedInstaller" to "BUILTIN\Administrators".
- The access permissions on it have changed (**red** = added permission; **green underlined** = unchanged permission; **red-lined-through** = removed permission).

To learn more about the syntax and structure of Linux and Windows file permissions, and also the color scheme used to show them here, see [Appendix: Interpreting File Access Permissions](#).

- Its content has changed, as shown by the changed value for the cryptographic signature of the file.

Based on your understanding of the file and what its ownership and permissions should be or have been, and whether or not the changes are suspicious, you may be able to infer whether this is a valid security issue that you need to address.

- If you think that this change is not a serious issue for this server at this time, either click Add Exception to temporarily hide the issue from future scan results, or click the file integrity policy name ("malware detection" in the

above example) to edit the policy to remove the target or add an exclusion to it.

For information, see [Specifying Exceptions](#) or [Using Patterns to Create Inclusions and Exclusions](#).

- If you want to inspect the baseline server that created the most recent baseline for the policy that triggered this issue, click the server name ("AMAZONA-0" in the above example) to view that server's summary page. Also, you can inspect the baseline report for that server (and other baseline servers, if there are more than one) on the file integrity policy's Edit page.

See [Act on FIM Issues, Findings, and Events](#), below.

Act on FIM Issues, Findings, and Events

Use FIM finding, issue, or event details to assess the security implications of a target change and take appropriate action.

- If the target modification occurred because the server undergoes frequent authorized modifications, and you want to stop monitoring it, you can remove the object's target or inclusion from your policy, or create an exclusion for the object within the target (see [Defining an Exclusion](#)).
- If the target change occurred because the object was added, modified, or removed legitimately and you want to start monitoring its new state, make the same modification to the baseline server and re-run the baseline scan.
- If the target modification occurred for an object that you want to keep monitoring, but the change is currently not a security issue—for example, if the object is undergoing testing or is scheduled for upgrade—create an [exception](#) to temporarily suppress reporting the event. No policy change or re-baselining is needed.

For information see [Specifying Exceptions](#), below.

- If the target change occurred because an object that you want to keep monitoring was modified, added or removed in error, just restore the previous state of the affected server(s). No policy change or re-baselining is needed.
- If none of the above conditions applies and it appears that the target's modification indicates potentially malicious activity on the server, stop using the server, quarantine it (if your organization's policy requires that), and immediately notify your incident response team or security forensics team.

Specifying Exceptions

When addressing file integrity scan results, you may decide not to act on (remediate) a particular finding or event at this time. If you do not wish to see the event repeated in future scans, you can classify it as an exception, which means that it will not appear in future scan results for a specified period of time.

Exceptions are useful for temporarily suppressing findings that you intend to correct eventually, but you would rather not be distracted by them until you have addressed more pressing file integrity issues.

To create a file integrity exception:

1. On the Halo portal, select a server group or server. Or, navigate to the Security Events History page and filter for file integrity events.
2. In the list of events, click Create Exception in the line for an event that you no longer want to see in scan results. (Or, select the checkboxes for one or more events, and then click Create Exception at the top of the page.) The Add Exception dialog box opens.

Add File Integrity Monitoring Exception

File Integrity Monitoring exceptions suppress reporting of file integrity monitoring violations for some period of time. Exceptions can be viewed and removed by choosing File Integrity Monitoring Exceptions from the Policies menu.

Exception For: /var/lib/rpm/__db.003, /var/lib/mlocate/mlocate.db, /var/lib/rpm/__db.001

Exception Expires In:

☐ 1 day
☐ 7 days
☒ 30 days
☐ Custom: (days)
☐ Never

Exception Applies to Policy(s):

☒ Policy: Affected policies
☐ All Policies

Exception Applies to Server(s):

☐ Server: Affected servers
☒ Group: Affected groups
☐ All Servers

Reason for Exception:

Configuration not final

Save Cancel

3. Use the selection buttons to choose:

- When the specified exception(s) should expire. Select or type a number of days, or select Never.
- Whether the exception should apply to events generated from the current policy only, or from all file integrity policies.
- Whether the exception should apply only to the server on which the event was reported, or to all servers in that server group, or to all servers in all server groups.







4. Enter a reason or explanation for creating the exception, then click Save.

The selected events and any associated alerts will no longer be reported, until the exception period ends or you re-baseline the policy. Running a baseline scan removes all exceptions from all of the policy's scanned servers.

To view or remove file integrity exceptions:

You can track and manage all of your file integrity exceptions from a single page in the Halo Portal:

1. Navigate to Policies > File Integrity Exceptions to view the File Integrity Exceptions page.

File Integrity Exceptions							
These files have violations that are suppressed in results reporting.							
Object	Policy Scope	Server Scope	OS Type	Expires ▼	Added By	Reason	Remove
/var/lib/rpm/__db.003	VG-web	Server: 漢字板名交じり文		2012-11-03 23:09	dice2@acmecloud.com	Changes frequently	
/var/lib/rpm/__db.001	All Policies	All Servers		2012-11-03 23:11	dice2@acmecloud.com	Changes frequently	
/var/lib/mlocate/mlocate.db	All Policies	Group: DB-UH7		2012-11-03 23:59	dice2@acmecloud.com	Unstable in this scan group	

The table includes one line for every defined exception, showing the target path, the policies/servers/groups that the exception applies to, when it expires, who created, it, and why.

2. Exceptions will automatically disappear from this list when they expire (or when you re-run a baseline scan). To explicitly remove an exception, click the Delete icon () in the line for that exception.

The exception will no longer appear on the File Integrity Exceptions page. Any future events involving that target will appear on the Security Events History page and in individual servers' scan results.



This section suggests some of the ways to efficiently deploy and configure Halo File Integrity Monitoring in small to very large-scale environments. The five steps below contain recommendations for designing server groups, creating policies, applying baselines, performing ongoing monitoring, and handling server upgrades in deployments that can scale from a few servers up to thousands of servers.

1. Organize servers into groups according to geography or function

The basic purpose of a server group is to hold similarly configured and purposed servers, so that the same set of Halo policies can be applied to all of them. In general, take these considerations into account when designing your groups.

- When setting up your server groups, especially in a very large deployment, Halo's nested groups to organize servers by type or geographic location—for example, "US -> East -> Web".
- If your deployment includes very similarly configured web servers but with different applications running on top of them, you combine hierarchical server groups with naming to clarify which policies should be applied to each group. For example, "US -> East -> Web-Servers -> Login" and "US -> East -> Web-Servers -> Messaging". Being specific and granular with groups and their names can make the assignment of policies and the generation of baselines simpler to manage.
- Servers within a given group should have the same function and very similar configurations. Usually, that means the same OS and the same applications. Often, the servers in a group are all clones of the same gold master image.
- You can have multiple versions of the same operating system or application in a single group. The multiple baselines feature of File Integrity Monitoring allows you to accommodate this kind of departure from absolute uniformity.
- You can also have more than one operating system platform in the same server group. File Integrity Monitoring allows you to assign both a Windows and a Linux policy to a single server group. Halo automatically applies the appropriate policy and its baselines to each server, depending on its platform.

Naming groups according to function complements the modular policy-design best practice described below, in which you create broad policies that cover the common operating systems and applications, plus additional focused policies for the specific applications or data that might be unique to certain subsets of servers.

2. Create narrowly targeted file integrity policies

When you create policies, you can use a Halo policy template's rules as a starting point or create a new policy that contains new rules. Whichever approach you choose, use the following principles to guide the customization of your file integrity policies.

Be *as specific as possible* when choosing your file and directory targets. For example:

- Be judicious about where to use recursion on a target directory. If you're concerned about the state of just a few files in a directory, it is more efficient to use them as separate specific targets, rather than setting up their parent directory for recursion.
- On the other hand, if you are more concerned about detecting the addition or removal of files inside a directory, marking the directory target for recursion will detect those additions or deletions (although it will also result in all existing files in the directory and its subdirectories being examined for changes.)
- We do not recommend targeting any files or directories that you expect to change dynamically over time, such as log files, temporary directories, upload directories or `pagefile.sys`.

For Linux policies:

- Monitor the operating system binary files in `/bin`, `/sbin`, `/usr/bin`, `/usr/sbin` and anywhere else appropriate if you have deployed code for your running services.
- Monitor specific configuration files (typically found in `/etc` or in its subdirectories).
- Monitor any static custom code or content that you have deployed to your servers and that should not change except when

you explicitly update it.

For Windows policies:

- Monitor the `Windows` and `System32` folders, and specific applications in the `Program Files` folder (as analogous to the operating system files in Linux).
- Monitor portions of `HKEY_LOCAL_MACHINE` in the Windows registry.
- Also monitor any static custom code or static content that you have deployed.

Since you can assign more than one file integrity policy to a server group, consider creating individual, modular file integrity policies for all of the various file types (OS-specific binary files, application-specific binary files, configuration files, static content, and so on) that you wish to monitor. It can be easier and more efficient to maintain these focused policies than it is to keep one monolithic policy up-to-date. And you can conveniently assemble them into a custom policy set for each server group, depending on its OS platform and its specific applications.

Note: Halo currently provides default OS-level file integrity policies for Windows Server platforms.

Assigning a policy to more than one server group

If you have several server groups with identical or very similar configurations, you might wish to assign the same file integrity policy to all of them. Instead, because of the way that file integrity target comparisons are made, it may be more efficient to clone the policy and rename it for each new group that you assign it to.

The reason for this is that, if you assign one file integrity policy to multiple groups, all baselines generated for that policy—from any of the groups—will be included in the scans of all of the groups. A large number of baselines that don't apply to a particular group would unnecessarily be included in the processing of that group's scans, leading to wasted time making the unneeded comparisons.

For information, in the *Halo Operations Guide* see [About Policy Inheritance](#).

3. Generate and actively manage baselines

Every unique server image used in your fleet requires a file integrity baseline that describes the configuration's canonical, secure state with respect to a given file integrity policy. If you wish to allow a single server group to accommodate more than one canonical image (for example, you might want it to support two or more upgrade levels of a given OS or application), you can do that by defining multiple baselines for the group's file integrity policy.

- Use multiple baselines for server groups whose software you are upgrading. For example, if you have three common "gold master" server images at different upgrade levels, from which you launch anywhere from a few to perhaps hundreds of servers in a given server group, you'll need to generate a baseline for each of those master images when editing the file integrity policy assigned to the server group.
- If you subsequently add new software to any server in that group to allow it to support a new function—or if you upgrade one to the master images and instantiate new servers from it—you'll then need to generate a baseline for that new configuration and assign it to the group's file integrity policy, so that future scans do not report the configuration as a (false positive) policy violation.
- Note that removing unused baselines is just as important as creating new ones. If you have an old baseline for a server that no longer reflects any of the machines running in your environment, leaving that baseline assigned to a policy is inefficient; it can lead to extra comparison time spent during a scan for no useful result. Make a point of manually (or automatically, through automation scripts) removing baselines from policies if the particular server configurations they represent are no longer running.

Alternatively, when you create each baseline, set its expiration so that it will expire shortly after you will have upgraded all of your servers to a new configuration. (You can do this if you upgrade on a regular schedule.) If you set a baseline's expiration to "never", you'll have to remember to manually delete it at the proper time.

- Creating modular, narrow policies and defining multiple baselines can go a long way toward reducing the potential "alert load" of false positives that can occur when software changes are introduced. By updating the individual affected policies (easier than searching for the affected targets in a monolithic policy) and generating new baselines, you can eliminate or greatly reduce those false positives across your fleet of servers.

Note: Adding and removing baselines in relation to software upgrades should be considered in the context of a "baseline lifecycle". See the discussion and diagram under [Step 5. Upgrade servers without disrupting file integrity scans](#).

Planning out your initial file integrity monitoring deployment ahead of time, including all unique server builds in your baselines, and modularizing your policies should help your ongoing operations run more smoothly.

Add comments to your baselines

When you create a baseline in the Halo portal, you should always put in a meaningful comment to help keep track of the baseline and remember what configuration it represents. At the very least include the date that the baseline was generated and the name of the user that created it. This helps capture the institutional knowledge and fosters coordination with the responsible party (or parties) as policies change, or as baselined servers are retired or removed from your environment.

Automate baselining

If you use an IT automation tool like Puppet, Chef, or Ansible in your environment, consider leveraging the Halo REST API in your scripts or recipes.

For example, if you have an automation script that handles things like new package installs or file content updates, have the script create a new file integrity baseline of the new or updated server configuration, so that the server group's file integrity policy can automatically adapt to the changed environment, instead of reporting false positives when the next file integrity scan is run.

4. Keep your ongoing monitoring efficient

To maintain maximum efficiency and effectiveness of your File Integrity Monitoring implementation, consider these recommendations for common practices and specific configuration settings that you can make. Making these settings adjustments is another way to significantly reduce your file integrity "event load" .

In the Site Administration > Scanner Settings tab you can configure Halo's behavior for the File Integrity Monitoring module (and others). The following are useful recommendations at any scale, but especially for large deployments:

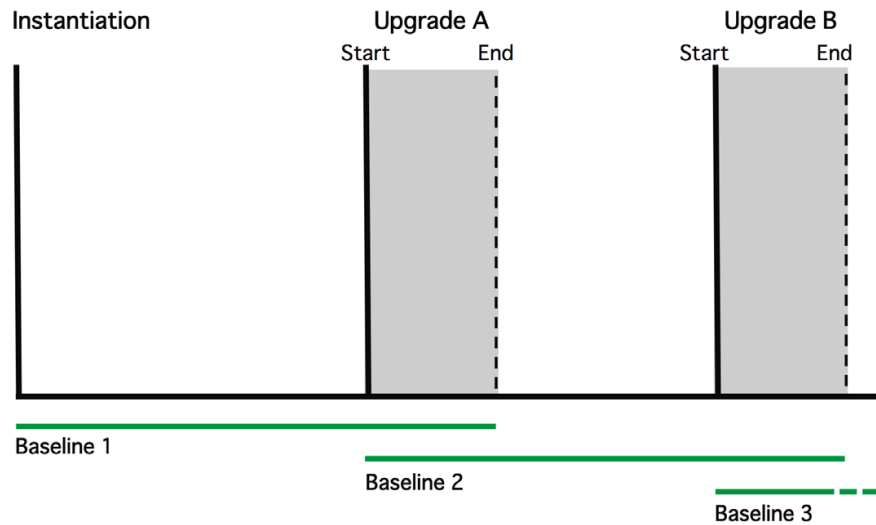
- **Scanning Frequency.** For maximum security during an investigation, you might temporarily set the file integrity scan frequency to as often as "hourly". For compliance monitoring, where immediate response is not as important, "daily" or even "weekly" might be sufficient. For general security purposes, "daily" for large installations and "twice daily" for smaller installations is probably a good compromise between maximum security and minimum load.

You can always run an ad-hoc or on-demand file integrity scan from the Halo portal or through the API at any time.

- **Scanning on agent Start.** If you are going to be launching a large number of servers regularly in your environment we recommend that you uncheck the checkbox "Execute scan on agent start" for the File Integrity Monitoring scanner. It's possible that a scan could start before your orchestration scripts have finished configuring the server, which could make that initial scan invalid.

5. Upgrade servers without disrupting file integrity scans

Depending on your organization's procedures for applying operating system and software patches and updates, you should take into consideration when and how often you should baseline or re-baseline the unique server configurations in your fleet, and when to expire older baselines. Plan your upgrades, baselining, and retirement of baselines to follow a consistent "baseline lifecycle" throughout your server fleet.



File integrity baseline lifecycle

A. If you re-instantiate servers from a patched gold master...

If your organization's policy is to perform updates only on a "gold master" image that you then use to instantiate all subsequent servers, you can

1. Launch that image and apply the updates.
2. Create a new baseline from that image. Do not remove the old baseline yet.

This is a useful preventative measure especially for large deployments, because the next file integrity scan could possibly begin before you have had a chance to upgrade all of your servers from the upgraded gold master. Having the old servers covered by the pre-existing baseline and the newly upgraded ones covered by the new baseline will prevent false alerts caused by the older configuration.

3. Shut down the existing, unpatched servers according to schedule and bring up the new, fully patched servers from the image that you have updated. You should not receive false-positive events from your file integrity scans.
4. Once the update is completed across all servers, the old baseline should automatically expire (if you have set its expiration date appropriately—typically 60 days or less if you apply updates regularly). Otherwise, manually remove the unneeded baselines.

B. If you patch your servers in-place...

If you will be upgrading systems in place, you should

1. Make sure that the first servers you upgrade each time are the ones you have built your file integrity baselines from.
2. Once those upgrades are complete, instead of just re-baselining the original baselined servers, create new, additional baselines for those upgraded servers—because the next file integrity scan could possibly begin before you have had a chance to upgrade all of your servers in place. Having the old servers covered by the pre-existing baseline and the newly upgraded ones covered by the new baseline will prevent false alerts caused by the older configuration.
3. Once all your servers are upgraded, the old baseline is no longer be needed and you can remove from the policies—or, if you have set its expiration appropriately, it will expire automatically. For example, if you update your servers every 60 days, set your baselines to expire at 60 days or sooner.

As previously mentioned, applying descriptive comments when creating a new baseline or re-baselining a server will help you keep everything clarified, making it easier to manage updates and baseline retirement.

Also keep in mind that you can automate the process of baselining and integrate it into your upgrade process by inserting Halo API client scripts into your server orchestration tools.

For More Information

For more information about how Halo File Integrity Monitoring works and how you can configure it to give you the best results, see the following discussions:

- [Setting up Server Groups](#) in the *Halo Operations Guide*
- [Creating a File integrity Policy](#) in this document.
- [Setting Up File Integrity Baselines](#) in this document.
- [Managing Ongoing Monitoring](#) in this document.

For information about using File Integrity Monitoring, see the [FIM FAQ](#) in our Support Forums.

Appendix: Interpreting File Access Permissions

Unexpected changes in a file or directory's access permissions can be indicative of an attack or intrusion. Halo displays the permissions for a changed file integrity target in a scan's event details, and also indicates whether the permissions themselves have changed. You can examine the permissions to determine whether a change to them is suspicious.

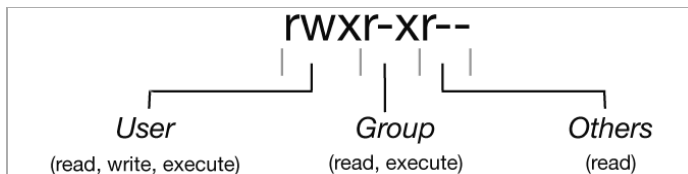
Linux Permissions

Every file and directory on a Linux system is owned by a user and by a group. Permissions for accessing that file are defined separately for the user, for the group, and for all others. "Other" is defined as a user that is not the owning user and is not a member of the owning group.

Linux supports three types of access permission:

- read. For a file, the ability to open it and read its contents. For a directory, the ability to list its contents.
- write. For a file, the ability to modify it (write new data to it). For a directory, the ability to add, remove, or rename files within the directory.
- execute. For a file, the ability to execute it as a program or script. For a directory, the ability to make it the current directory (as with the `cd` shell command), and the ability to access files within it.

Halo displays a Linux file or directory's permissions using the symbolic mode common to shell displays, with three sets of three single character symbols, specifying the permissions of the owner, group, and others, respectively. For example:



In this example, the user owner has full permissions on the file, the owning group can read and execute the file but not modify it, and others can open and read the file, but not modify or execute it.

Special Linux permissions. The above permissions string can include several additions or substitutions in special cases:

- Directory. If the file is a directory, the permissions string may be prefixed with `d`. For example:

`drwxr-xr--`

- setuid /setgid bit. If the executable file's setuid or setgid bit is set (meaning that the file will execute as the user owner or group owner, with the owner's permissions), `s` or `S` is substituted for `r` in the "User" or "Group" permission. For example:

`swxr-xr--` or `rwxs-xr--`

- Sticky bit. If a directory's sticky bit is set (meaning that no users except the directory's owner and the superuser can

rename or delete files within the directory), t or T is substituted for x in the "Others" (all users) permission. For example:

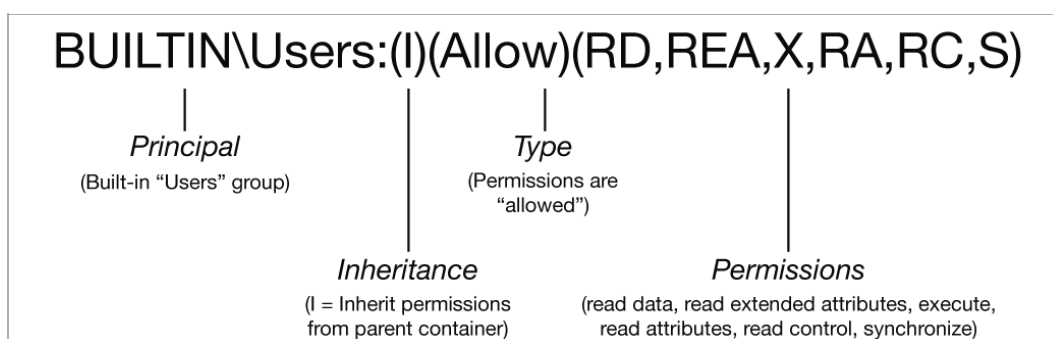
`drwxr-xr-t`

Windows Access Control Entries

All Windows securable objects such as files and registry keys have *security descriptors*, which allow the system to determine whether access should be granted to the object. A security descriptor identifies the object's owner and contains, among other things, a discretionary access control list (ACL).

The ACL is an ordered list of zero or more access control entries (ACEs). Each ACE specifies the type of access being addressed (allow, deny, audit), includes inheritance flags, identifies the principal (user or group) for whom this access is specified, and contains an access mask that lists the operations that are allowed or denied to that principal. The access mask is represented as a 32 bit string; each bit location in the access mask represents a specific type of permission for the object.

In file-integrity scan details, Halo represents an object's ACL as a sequence of text lines, one for each ACE—that is, one for each principal with defined permissions on the object. In each ACE, Halo replaces the non-human-readable ACE bit mask with a series of multi-character codes specifying the permission details:



These are values you might see for each of the ACE elements:

- **Principal.** This is the name of the user or group owner, preceded by a scope designation. The scope may be NT AUTHORITY, BUILTIN, the local domain (machine name), or Active Directory/LDAP domain.

For local user accounts where the scope is the machine name, Halo replaces the machine name with "Local" so that events will not be created when local accounts on different machines differ only by the machine name.

- **Inheritance.** The inheritance strings can have these values and meanings:
 - OI. Object inherit (applies only to folders and non-leaf registry keys)
 - CI. Container inherit (applies only to folders and non-leaf registry keys)
 - IO. Inherit only (applies only to folders and non-leaf registry keys)
 - NP. Do not propagate the inherit (applies only to folders and non-leaf registry keys)
 - I. Access control is inherited from parent container

It is also possible for a file object to have no inheritance designation in its ACE.

- **Type.** The type of the access control entry can be either Allow or Deny.
- **Permissions.** The following are the rights that can appear in the permissions string in the Halo portal. Note that Halo displays only "specific rights," not the "simple rights" groupings that you may find using icacls or a similar tool.

File or folder:

- AD. Append data/add subdirectory
- AS. Access system security
- DC. Delete child
- DE. Delete
- GA. Generic all
- GE. Generic execute
- GR. Generic read
- GW. Generic write

Registry key:

- CC. Create child
- CR. Control access
- DC. Delete child
- DT. Delete tree
- KA. Key all
- KR. Key read
- KW. Key write
- KX. Key execute

- MA. Maximum allowed
- RA. Read attributes
- RC. Read control
- RD. Read data/list directory
- REA. Read extended attributes
- S. Synchronize
- WA. Write attributes
- WD. Write data/add file
- WDAC. Write DAC
- WEA. Write extended attributes
- WO. Write owner
- X. Execute/traverse
- LC. List children
- LO. List object
- RC. Read control
- RP. Read property
- SD. Standard delete
- SW. Self write
- WD. Write DAC
- WO. Write owner
- WP. Write property

Interpreting Colored Elements in a File Integrity Issue or Event

In the more details display of a file integrity finding or event, colors indicate the nature of changed elements:

- Any file signature or critical metadata value that does not match the baseline is highlighted in red.
- Items that have been removed since the baseline scan are highlighted in red and lined-through.
- Items that have been added since the baseline scan are highlighted in green.

Note that if permissions are added to or removed from a Windows object, the Permissions field in scan details may display other items twice—once in lined-through red and once in green. This occurs because Halo examines the list in order from the top, and only items that have not changed in content or position remain black (unhighlighted) in the scanned object details. Any item whose position has shifted will appear twice, in green at its new position and in red at its old position.

Permissions
Local\R9:(Allow)(RD,REA,X,RA,RC,S) NT AUTHORITY\SYSTEM:(I)(Allow) (RD,WD,AD,REA,WEA,X,DC,RA,WA,DE,RC,WDAC,WO,S) BUILTIN\Administrators:(I)(Allow) (RD,WD,AD,REA,WEA,X,DC,RA,WA,DE,RC,WDAC,WO,S) BUILTIN\Users:(I)(Allow)(RD,REA,X,RA,RC,S)
<u>Local\QA:(Allow)(RD,REA,X,RA,RC,S)</u> NT AUTHORITY\SYSTEM:(I)(Allow) (RD,WD,AD,REA,WEA,X,DC,RA,WA,DE,RC,WDAC,WO,S) BUILTIN\Administrators:(I)(Allow) (RD,WD,AD,REA,WEA,X,DC,RA,WA,DE,RC,WDAC,WO,S) BUILTIN\Users:(I)(Allow)(RD,REA,X,RA,RC,S) Local\R9:(Allow)(RD,REA,X,RA,RC,S)

For example, in the scan details above, the first item in the list ("local\R9:...") was in the baseline scan but was replaced on this server with a different permission ("Local\QA:...") before the most recent scan. The remaining 3 items did not change their content or position in the list, so they remain black. The added item appears underlined in green at its appropriate point in the list, and the removed item appears after the list, lined-through in red.

Permissions
Local\I9:(Allow)(RD,REA,X,RA,RC,S) NT AUTHORITY\SYSTEM:(I)(Allow) (RD,WD,AD,REA,WEA,X,DC,RA,WA,DE,RC,WDAC,WQ,S) BUILTIN\Administrators:(I)(Allow) (RD,WD,AD,REA,WEA,X,DC,RA,WA,DE,RC,WDAC,WQ,S) BUILTIN\Users:(I)(Allow)(RD,REA,X,RA,RC,S)
<u>NT AUTHORITY\SYSTEM:(I)(Allow)</u> <u>(RD,WD,AD,REA,WEA,X,DC,RA,WA,DE,RC,WDAC,WQ,S)</u> <u>BUILTIN\Administrators:(I)(Allow)</u> <u>(RD,WD,AD,REA,WEA,X,DC,RA,WA,DE,RC,WDAC,WQ,S)</u> <u>BUILTIN\Users:(I)(Allow)(RD,REA,X,RA,RC,S)</u> Local\I9:(Allow)(RD,REA,X,RA,RC,S) NT AUTHORITY\SYSTEM:(I)(Allow) (RD,WD,AD,REA,WEA,X,DC,RA,WA,DE,RC,WDAC,WQ,S) BUILTIN\Administrators:(I)(Allow) (RD,WD,AD,REA,WEA,X,DC,RA,WA,DE,RC,WDAC,WQ,S) BUILTIN\Users:(I)(Allow)(RD,REA,X,RA,RC,S)

In another example of scan details (above), the first item in the list ("local\I9:...") was in the baseline scan but was removed from this server before the most recent scan. The remaining 3 items have changed position—what was item two is now item one, and so on. Therefore the entire original list appears to have been removed (and is lined-through in red), and the server's entire current list appears to have been added (and is underlined in green).