Understanding App Control event tags

Article • 10/22/2024 • 3 contributors •

Applies to: Windows 11, Windows 10

Feedback

In this article

App Control for Business events include many fields, which provide helpful troubleshooting information to figure out exactly what an event

means. This article describes the values and meanings for a few useful event tags.

SignatureType

Represents the type of signature which verified the image.

Expand table

SignatureType Value	Explanation
0	Unsigned or verification hasn't been attempted
1	Embedded signature
2	Cached signature; presence of a CI EA means the file was previously verified
3	Cached catalog verified via Catalog Database or searching catalog directly
4	Uncached catalog verified via Catalog Database or searching catalog directly
5	Successfully verified using an EA that informs CI that catalog to try first
6	AppX / MSIX package catalog verified
7	File was verified

Requested and Validated Signing Level

Represents the signature level at which the code was verified.

Expand table

SigningLevel Value	Explanation
0	Signing level hasn't yet been checked
1	File is unsigned or has no signature that passes the active policies
2	Trusted by App Control for Business policy
3	Developer signed code
4	Authenticode signed
5	Microsoft Store signed app PPL (Protected Process Light)
6	Microsoft Store-signed
7	Signed by an Antimalware vendor whose product is using AMPPL
8	Microsoft signed
11	Only used for signing of the .NET NGEN compiler
12	Windows signed
14	Windows Trusted Computing Base signed

VerificationError

Represents why verification failed, or if it succeeded.

Expand table

VerificationError Value	Explanation
0	Successfully verified signature.
1	File has an invalid hash.

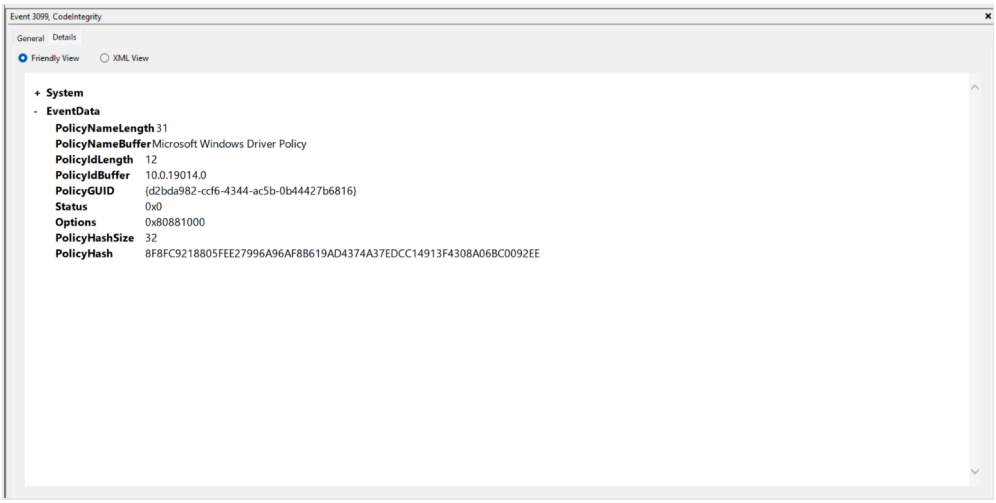
2	File contains shared writable sections.
3	File isn't signed.
4	Revoked signature.
5	Expired signature.
6	File is signed using a weak hashing algorithm, which doesn't meet the minimum policy.
7	Invalid root certificate.
8	Signature was unable to be validated; generic error.
9	Signing time not trusted.
10	The file must be signed using page hashes for this scenario.
11	Page hash mismatch.
12	Not valid for a PPL (Protected Process Light).
13	Not valid for a PP (Protected Process).
14	The signature is missing the required ARM processor EKU.
15	Failed WHQL check.
16	Default policy signing level not met.
17	Custom policy signing level not met; returned when signature doesn't validate against an SBCP-defined set of certs.
18	Custom signing level not met; returned if signature fails to match `CISigners` in UMCI.
19	Binary is revoked based on its file hash.
20	SHA1 cert hash's timestamp is missing or after valid cutoff as defined by Weak Crypto Policy.
21	Failed to pass App Control for Business policy.

22	Not Isolated User Mode (IUM) signed; indicates an attempt to load a standard Windows binary into a virtualization-based security (VBS) trustlet.
23	Invalid image hash. This error can indicate file corruption or a problem with the file's signature. Signatures using elliptic curve cryptography (ECC), such as ECDSA, return this VerificationError.
24	Flight root not allowed; indicates trying to run flight-signed code on production OS.
25	Anti-cheat policy violation.
26	Explicitly denied by App Control policy.
27	The signing chain appears to be tampered/invalid.
28	Resource page hash mismatch.

Policy activation event Options

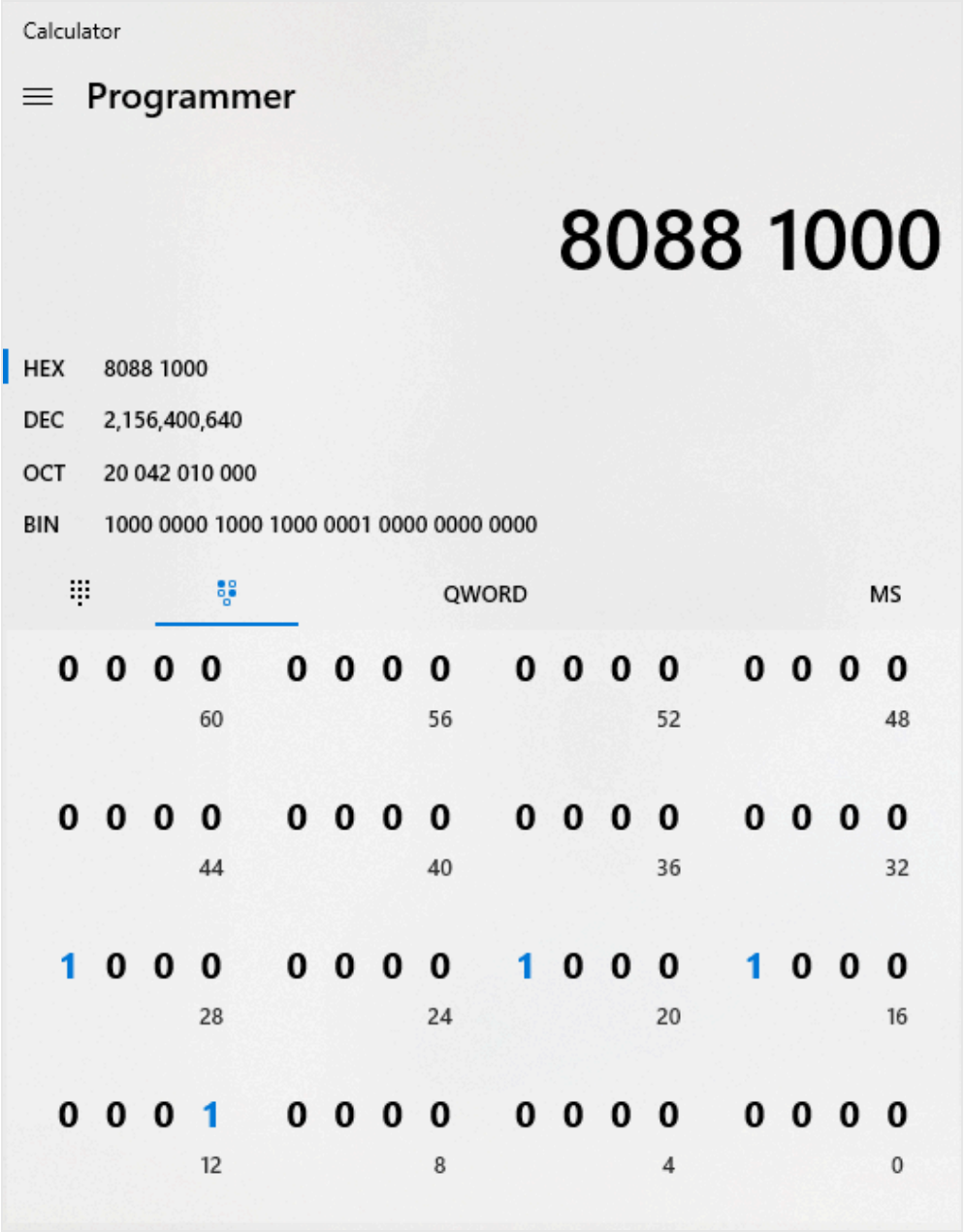
The App Control policy rule option values can be derived from the "Options" field in the Details section for successful policy activation events. To parse the values, first convert the hex value to binary. To derive and parse these values, follow the below workflow.

- Access Event Viewer.
- Access the Code integrity 3099 event.
- Access the details pane.
- Identify the hex code listed in the "Options" field.
- Convert the hex code to binary.



For a simple solution for converting hex to binary, follow these steps:

1. Open the Calculator app.
2. Select the menu icon.
3. Select **Programmer** mode.
4. Select **HEX**.
5. Enter your hex code. For example, `80881000`.
6. Switch to the **Bit Toggling Keyboard**.



This view provides the hex code in binary form, with each bit address shown separately. The bit addresses start at 0 in the bottom right. Each bit address correlates to a specific event policy-rule option. If the bit address holds a value of 1, the setting is in the policy.

Next, use the bit addresses and their values from the following table to determine the state of each policy rule-option. For example, if the bit address of 16 holds a value of 1, then the **Enabled: Audit Mode (Default)** option is in the policy. This setting means that the policy is in audit mode.

Expand table

Bit Address	Policy Rule Option
2	`Enabled:UMCI`
3	`Enabled:Boot Menu Protection`
4	`Enabled:Intelligent Security Graph Authorization`
5	`Enabled:Invalidate EAs on Reboot`
7	`Required:WHQL`
10	`Enabled:Allow Supplemental Policies`
11	`Disabled:Runtime FilePath Rule Protection`
13	`Enabled:Revoked Expired As Unsigned`
16	`Enabled:Audit Mode (Default)`
17	`Disabled:Flight Signing`
18	`Enabled:Inherit Default Policy`
19	`Enabled:Unsigned System Integrity Policy (Default)`
20	`Enabled:Dynamic Code Security`
21	`Required:EV Signers`
22	`Enabled:Boot Audit on Failure`
23	`Enabled:Advanced Boot Options Menu`
24	`Disabled:Script Enforcement`
25	`Required:Enforce Store Applications`
27	`Enabled:Managed Installer`
28	`Enabled:Update Policy No Reboot`

Microsoft Root CAs trusted by Windows

The Microsoft Root certificates can be allowed and denied in policy using 'WellKnown' rules. The mapping between the root's ASN1 encoded RSA PKCS#1 public key and the WellKnown values, expressed in hexidecimal, are listed below

Expand table

Root ID	Root Name	Root Public Key
0	None	N/A
1	Unknown	N/A
2	Self-Signed	N/A
3	Microsoft Authenticode(tm) Root Authority	`30820122300D06092A864886F70D0101010500038201`
4	Microsoft Product Root 1997	`30820122300D06092A864886F70D0101010500038201`
5	Microsoft Product Root 2001	`30820222300D06092A864886F70D0101010500038202`
6	Microsoft Product Root 2010	`30820222300D06092A864886F70D0101010500038202`
7	Microsoft Standard Root 2011	`30820222300D06092A864886F70D0101010500038202`
8	Microsoft Code Verification Root 2006	`30820222300D06092A864886F70D0101010500038202`

9	Microsoft Test Root 1999	`3081DF300D06092A864886F70D0101010500038 1CD003`
0A	Microsoft Test Root 2010	`30820222300D06092A864886F70D01010105000382020`
0B	Microsoft DMD Test Root 2005	`30820122300D06092A864886F70D01010105000382010`
0C	Microsoft DMDRoot 2005	`30820222300D06092A864886F70D01010105000382020`
0D	Microsoft DMD Preview Root 2005	`30820222300D06092A864886F70D01010105000382020`
0E	Microsoft Flight Root 2014	`30820222300D06092A864886F70D01010105000382020`
0F	Microsoft Third Party Marketplace Root	`30820222300D06092A864886F70D01010105000382020`
14	Microsoft Trusted Root Store	N/A
15	Microsoft OEM Root Certificate Authority 2017	`30820222300D06092A864886F70D01010105000382020`
16	Microsoft Identity Verification Root Certificate Authority 2020	`30820222300D06092A864886F70D01010105000382020`

For well-known roots, the TBS hashes for the certificates are baked into the code for App Control for Business. For example, they don't need to be listed as TBS hashes in the policy file.

Status values

Represents values that are used to communicate system information. They are of four types: success values, information values, warning values, and error values. See NTSATUS for information about common usage details.

Feedback

Was this page helpful? Yes No

Provide product feedback