



We use optional cookies to improve your experience on our websites, such as through social media connections, and to display personalized advertising based on your online activity. If you reject optional cookies, only cookies necessary to provide you the services will be used. You may change your selection by clicking "Manage Cookies" at the bottom of the page. [Privacy Statement](#) [Third-Party Cookies](#)

Accept

Reject

Manage cookies



Learn

Discover ▾

Product documentation ▾

Development languages ▾

Topics ▾



Sign in

Windows App Development

Explore ▾

Development ▾

Platforms ▾

Troubleshooting

More ▾

Dashboard



Mandatory Integrity Control

Article • 03/25/2021 • 7 contributors

[Feedback](#)

In this article

[Integrity Labels](#)

[Process Creation](#)

[Mandatory Policy](#)

Mandatory Integrity Control (MIC) provides a mechanism for controlling access to securable objects. This mechanism is in addition to discretionary access control and evaluates access before access checks against an object's [discretionary access control list](#) (DACL) are evaluated.

MIC uses integrity levels and mandatory policy to evaluate access. [Security principals](#) and securable objects are assigned integrity levels that determine their levels of protection or access. For example, a principal with a low integrity level cannot write to an object with a medium integrity level, even if that object's DACL allows write access to the principal.

Windows defines four integrity levels: low, medium, high, and system. Standard users receive medium, elevated users receive high. Processes you

start and objects you create receive your integrity level (medium or high) or low if the executable file's level is low; system services receive system integrity. Objects that lack an integrity label are treated as medium by the operating system; this prevents low-integrity code from modifying unlabeled objects. Additionally, Windows ensures that processes running with a low integrity level cannot obtain access to a process which is associated with an app container.

Integrity Labels

Integrity labels specify the integrity levels of securable objects and security principals. Integrity labels are represented by *integrity SIDs*. The integrity SID for a securable object is stored in its *system access control list* (SACL). The SACL contains a **SYSTEM_MANDATORY_LABEL_ACE** *access control entry* (ACE) that in turn contains the integrity SID. Any object without an integrity SID is treated as if it had medium integrity.

The integrity SID for a security principal is stored in its access token. An access token may contain one or more integrity SIDs.

For detailed information about the defined integrity SIDs, see [Well-known SIDs](#).

Process Creation

When a user attempts to launch an executable file, the new process is created with the minimum of the user integrity level and the file integrity level. This means that the new process will never execute with higher integrity than the executable file. If the administrator user executes a low integrity program, the token for the new process functions with the low integrity level. This helps protect a user who launches untrustworthy code from malicious acts performed by that code. The user data, which is at the typical user integrity level, is write-protected against this new process.

Mandatory Policy

The [SYSTEM_MANDATORY_LABEL_ACE](#) ACE in the SACL of a securable object contains an access mask that specifies the access that principals with integrity levels lower than the object are granted. The values defined for this access mask are [SYSTEM_MANDATORY_LABEL_NO_WRITE_UP](#), [SYSTEM_MANDATORY_LABEL_NO_READ_UP](#), and [SYSTEM_MANDATORY_LABEL_NO_EXECUTE_UP](#). By default, the system creates every object with an access mask of [SYSTEM_MANDATORY_LABEL_NO_WRITE_UP](#).

Every access token also specifies a mandatory policy that is set by the [Local Security Authority](#) (LSA) when the token is created. This policy is specified by a [TOKEN_MANDATORY_POLICY](#) structure associated with the token. This structure can be queried by calling the [GetTokenInformation](#) function with the value of the *TokenInformationClass* parameter set to [TokenMandatoryPolicy](#).


Feedback

Was this page helpful?



 Yes

 No

[Provide product feedback](#)  | [Get help at Microsoft Q&A](#)

 English (United States)

  Your Privacy Choices


 Theme 

[Manage cookies](#)


[Previous Versions](#)

[Blog](#) 

[Contribute](#)

[Privacy](#) 

[Terms of Use](#)

[Trademarks](#) 

© Microsoft 2024