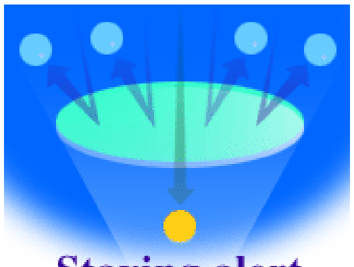


Staying alert with Execution Control Lists

by Amy E. Smith
(with Charlie Kaufman, Chuck Bassett, and Mary Ellen Zurko)

Level: Intermediate
Works with: Domino 5.0
Updated: 12/01/99

Related links:
Domino 5 Administration Help
Recommendations for deploying tighter ECLs
5.0.2 Release Notes
Sidebar: ECL risk levels

Get the PDF:

Chances are, you've never thought much about ECLs, mainly because you've never had to. Now, as a result of changes in 5.0.2, ECLs are going to be making themselves known to system administrators and their users. This is a good thing, because it presents a valuable opportunity for administrators to think about the role that ECLs play in their workplace, and to implement (or in many cases, re-deploy) them accordingly.

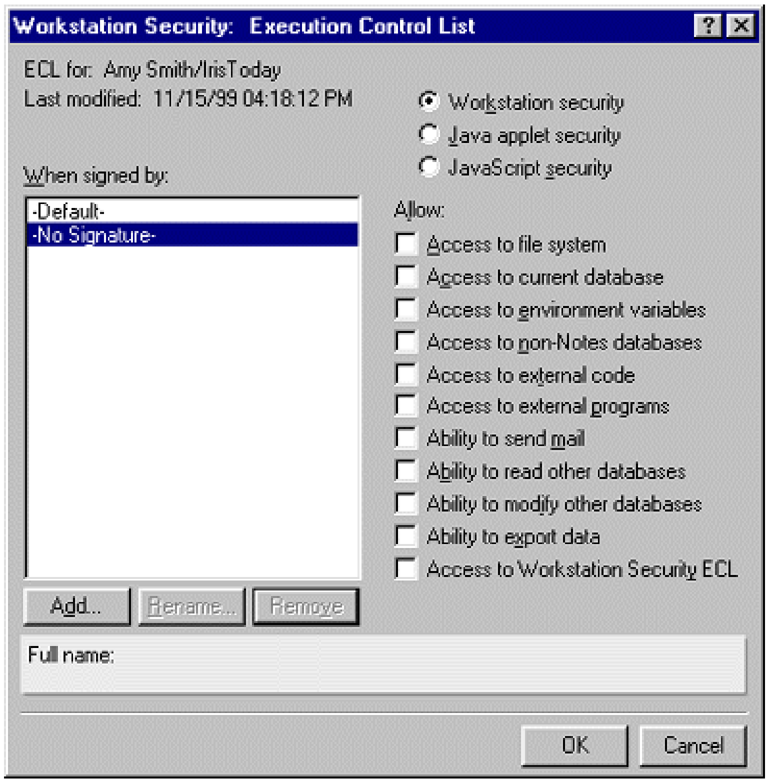
The Execution Control List (ECL) is a potentially powerful part of the system administrator's security toolbox, yet it is frequently under-utilized at best, and overlooked at worst. Waiting quietly in the background on every client workstation, like a watchdog, the ECL is designed to protect user workstations against code from unknown or suspect sources. The ECL determines whether the signer of the code is allowed to have its code run on a given workstation, and defines the extent to which the code has access to various workstation functions and is gated by the workstation security ECL.

In this article, you will learn how ECLs work, about their importance in user workstation security, and how you, as a system administrator, can deploy and manage them effectively in your workplace.

For the purposes of this article, the term "active content" is used to refer to items that are verified and screened by the ECL. This includes formulas, scripts, agents, design elements in databases and templates, documents with stored forms, actions, buttons, hot spots, as well as malicious code (such as viruses and Trojan horses) -- in short, anything that can be executed on a user workstation.

How ECLs work

ECLs list trusted authors of active content. In Notes, database design elements, formulas, scripts, and other active content are signed with the ID of the user who created it or last modified it. In order for active content to be trusted, and thereby allowed to run on the workstation, the signer must be listed in the ECL.



ECL for: Amy Smith/IrisToday
Last modified: 11/15/99 04:18:12 PM

[✓ Workstation security] Java applet security JavaScript security

When signed by:
- -Default-
- -No Signature-

Allow:
- ☐ Access to file system
- ☐ Access to current database
- ☐ Access to environment variables
- ☐ Access to non-Notes databases
- ☐ Access to external code
- ☐ Access to external programs
- ☐ Ability to send mail
- ☐ Ability to read other databases
- ☐ Ability to modify other databases
- ☐ Ability to export data
- ☐ Access to Workstation Security ECL

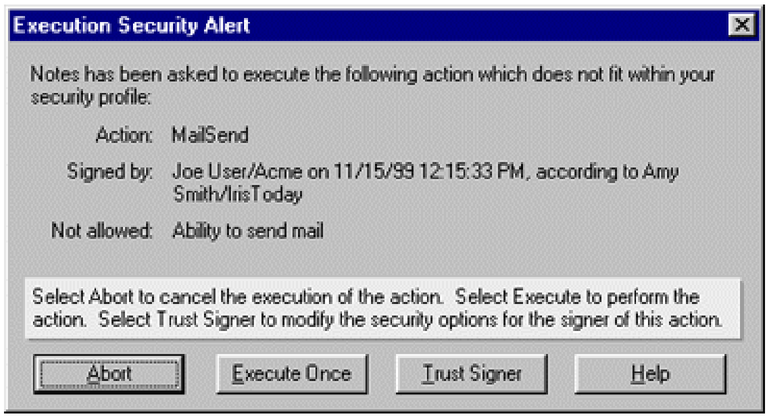
Full name:

For each signer listed in the ECL, workstation security settings can be enabled for access to protected operations, such as the ability to access the workstation file system or external programs. For a description of the workstation security options, see the **Workstation access options sidebar**. (Note: Although this article concentrates specifically on workstation security ECLs, descriptions of Java and JavaScript security ECL options are also provided in the sidebar.)

Note the list of signers in the ECL dialog shown above. You can see that the "No Signature" entry (highlighted) does not have any workstation security options enabled.

When active content runs on a user workstation and attempts a potentially harmful operation, several things happen. Notes verifies the code is signed, looks up the signer of the code in the client's ECL, and then checks the signer's ECL settings to determine whether the action is allowed. If the signer of the code is listed in the client's ECL and the appropriate setting is enabled, the code is executed.

If the active content attempts an action that has not been enabled for its particular signer in the ECL, or if the signer is not listed in the ECL, an Execution Security Alert (ESA) is generated. The ESA specifies the attempted action, the item's signer, and the ECL access option that is not allowed.


Execution Security Alert

Notes has been asked to execute the following action which does not fit within your security profile:

Action: MailSend

Signed by: Joe User/Acme on 11/15/99 12:15:33 PM, according to Amy Smith/IrisToday

Not allowed: Ability to send mail

Select Abort to cancel the execution of the action. Select Execute to perform the action. Select Trust Signer to modify the security options for the signer of this action.

When users see an ESA, they have three options:

- **Abort** -- Cancel the execution of the action in question.
- **Execute Once** -- Perform the action, but doing so does **not** modify the ECL configuration. If the same action is attempted by the same signer in the future, the ESA appears again.
- **Trust Signer** - Performs the action for the signer **and** modifies the ECL configuration, adding permission for the signer to execute the action anytime.

The ESA shown above was generated on a workstation that uses the ECL options shown earlier. The active content is this case is a mail message that includes a button that perfoms a Mail Send. Note that while the active content is signed, the signer is not trusted in the ECL so the action is disallowed. (The "No Signature" entry in the ECL signer list covers both unsigned code and code that is signed by an identity or organization that can't be authenticated.). If the user were to click "Trust Signer," the signer would be added to the ECL, and the action would be enabled for that signer.

Trusting unsigned content is extremely risky, and creates a security hole that allows potentially harmful code, malicious or otherwise, to access user workstations. Trusting signed active content from other organizations is also risky, as merely having a signature doesn't make an item trusted. Before adding an active content author to your ECL, you must decide if you trust the author has created safe code.

Propagating ECLs in the workplace

There are two kinds of ECLs: the Administration ECL, which resides in the Domino Directory (names.nsf), and the workstation ECL, which is stored in the workstation's Desktop file (desktop.dsk or desktop5.dsk). In most cases, the Administration ECL is the template for all workstation ECLs. During the installation of the first server in the domain, the Administration ECL is created with default settings. Subsequently, whenever a new client is set up, a copy of the Administration ECL is created locally on the user workstation. The current Notes user ID is also added to the local ECL, with all access allowed. For example, when John Doe's Notes client is being set up, John Doe is automatically added to the client ECL signer list. If the home server is unavailable at setup time, such as when a user is disconnected, a default ECL is created.

ECLs are not static. They are designed to be reconfigured to meet changing security requirements. Administration ECLs can be edited through the Domino Administrator client. There are several ways to update user ECLs, by:

- Editing the user ECL dialog.
- Clicking Trust Signer (although this is not always desirable; see below).
- Refreshing with an updated version of the Administration ECL. (For a

description of this procedure, see **Recommendations for deploying tighter ECLs** in the **5.0.2 release notes**.

ECL changes for 5.0.2

Until release 5.0.2 of Notes/Domino, ECLs and signatures were provided as tools for administrators and users to implement as security policy dictated. In release 5.0.2, IBM/Lotus began concentrating efforts on fine-tuning ECLs to provide the optimum balance between security and usability.

ECL default settings

A major change in 5.0.2 is the change in the ECL default settings. Previously, default ECL settings favored a more open configuration. They enabled **all** access options for the following signatures:

- **Default** -- Trusts code signed with any signature
- **No Signature** - Trusts unsigned or unauthenticated code
- **(UserName)** - Trusts code signed with the user's ID (user ID that was added when the client ECL was first set up)
- **Lotus Notes Template Development** - All Notes templates are signed with this ID, and this signature is trusted by default

If administrators failed to supply a Administration ECL with different settings, users would not get any ESAs; however, this meant that workstation security was, for all intents and purposes, nonexistent.

For release 5.0.2, the default settings are now "tight" instead of open, meaning that the access options for signatures not known to be trustworthy have been disabled. The new default ECL settings do not allow access to protected operations for unsigned or untrusted formulas and code. Consequently, secure ECL defaults are implemented for new domain and client installations, as well as for domains that never modified their original default Administration ECL.

Note that the secure ECL defaults are applied automatically only during setup of new client ECLs. To implement the secure defaults for existing (pre-5.0.2) clients, Administration ECLs should be updated with the secure settings and the @RefreshECL function can be used to "push" updated Administration ECLs to existing clients.

When using the new default ECLs in 5.0.2, users will be seeing ESAs with far greater frequency than ever before. Both active content that is signed and trustworthy, and that with untrusted or no signatures, will produce warnings unless remedial action is taken, either by updating the Administration ECL or clicking "Trust Signer."

It's often tempting for users to just click "Trust Signer" every time they get an ESA. The problem with this is that, as the ECL is modified, it becomes more and more open and allows greater access to code that attempts to execute on the user workstation. This can inadvertently create security risk, especially if unsigned code is trusted. This problem can be offset through careful ECL planning. Signing all custom databases and templates with IDs included in the Administration ECL, and then refreshing user ECLs, will tighten security and minimize ESAs (and user annoyance).

Administrators can also reset the ECL to disable all workstation protection (in effect, restore the pre-5.0.2 defaults) before deploying end-user ECLs during client setup. This means that users would stop getting ESAs, as restoring the default settings has the same effect as allowing users to always "Trust Signer." Users can also edit their ECLs, once the client has been setup, to restore the pre-5.0.2 default settings. In both cases, however, this leaves user workstations open to potential security problems.

Execution Security Alert
There was a small user interface change made to the Execution Security Alert for 5.0.2, as well. Prior to 5.0.2, if users opted to Trust Signer, they were also prompted to trust a signer's entire organization. This option was removed in 5.0.2, because while it might be necessary to trust a signer in order to run something on the workstation, it is not necessary to enable the same options for the signer's entire organization.

Signing design elements
Lastly, for 5.0.2, most design elements that have executable code associated with them (for example, buttons, fields, formulas) can be signed and have their signatures checked at time of execution (for example, when a button on a form is clicked). This enhancement, which remedies several reported bugs, makes sure that an organization running a tight ship can associate code with any of the many options available in Designer, and not worry about users needing to leave a hole in their protection by granting "no signature" any access rights.

See the **5.0.2 release notes** for specific information about these enhancements.

Signature policies: proactive ECL management

The changes in 5.0.2 present an opportunity for administrators to rethink their ECL strategies and to be proactive in managing and deploying ECLs in their organization. An excellent way to do this is through the use of signature policies. A signature policy is essentially a system for administrators to plan for, and configure in the ECL, those signatures that are trusted to sign active content, those that are not, and to what extent the trusted signatures can access protected workstation operations. Not only does a signature policy promote sound security practices, but ideally, it minimizes or negates the need for users to deal with ECLs.

Implementing a signature policy in your organization requires some time investment on the part of both the administrator and the organization; there is maintenance overhead for such tasks as centralizing signing, keeping administration and workstations ECLs updated, and so on. However, the benefits to be realized are significant.

First, it is good information systems practice. ECLs protect user workstations from problems caused by active content, malicious or otherwise. It's possible to be exposed to code that was written with no malicious intent, but can still do damage because of coding errors. Setting up safeguards through a signature policy, such as only trusting certain users to sign/write code, reduces your exposure to both malicious and buggy code, and minimizes down time and support calls.

Second, it is good administrative practice. Having a signature policy in place reduces the chances of making mistakes (such as trusting an unsigned formula), compared to when signatures are trusted *ad hoc*, such as when users react to ESAs. In addition, the existence of a signature policy is frequently a good vehicle for setting down end-user security policy and practices.

Lastly, it is good business practice. A well-implemented signature policy works in tandem with corporate security practices to protect corporate information assets. It encourages a conscious approach to enabling access to those assets.

There are two strategies to think about when considering a signature policy:
- Managing and deploying user ECLs
- Trusting active content

Managing user ECLs
There are several options for managing and deploying user ECLs that range from minimal to maximum security, and may or may not require the implementation of a signature policy. Whether and how you decide to implement a signature policy in your organization depends on several factors; namely, the time and effort required for maintaining it, size and sophistication of the user community, nature of the business, and the extent to which users communicate externally.

One way to manage ECLs is by not managing them. This is the least secure method of all. User ECLs are set to pre-5.0.2 defaults, so that everyone, even unidentified signers, is trusted. User impact is minimal, since, as a result, users will never get ESAs. So, while you as an administrator will rarely be bothered by someone who needs to have their ECL updated, there is a greater risk for damage by malicious code. This kind of scenario is appropriate in organizations with small user communities that have physical security and no connections to the outside world.

The next, more secure option for managing ECLs is the "*ad hoc* trusting" method, where who to trust is determined by examining what ESAs arise in regular use, and users are instructed by their system administrator about who to trust. As these decisions are made, the Administration ECL is updated, and user ECLs are refreshed accordingly. (See the **5.0.2 release notes** for detailed instructions about *ad hoc* trusting).

The next couple of ECL management strategies require the use of signature policies. The first, which manages to incorporate a high degree of security and flexibility, relies on a set of policies and procedures. It includes guidelines for who is be trusted and who is not. There are procedures for keeping the Administration ECL up-to-date, and refreshing user ECLs regularly as the Administration ECL is updated. Users are given clear instructions for reporting ECL warnings, and there are firm policies about never trusting signers *ad hoc*, or clicking "execute once." Consequently, when ESAs do occur, it is either because of a mistake -- for instance, someone distributed code using a non-approved ID, or a database design element happens to be unsigned -- or because it is an actual security problem.

The most stringent signature policy is that which does not allow users to modify their ECLs. This means that they cannot edit their own workstation ECL, nor can they run unsigned or disallowed code. Should they get an ESA, the only option is to abort the operation. Administrators can set this option in the Administration ECL, by disabling the "Allow users to modify" option. When the Administration ECL is copied to user workstations, the option disallows users from editing their ECLs. This type of signature policy works best for companies in which users run a small, tightly controlled set of applications.

Trusting active content
An important aspect of a signature policy is defining a methodology for trusting signers, which takes into account signed content that comes from both within and outside the organization.

For active content that comes from external sources (for example, third-party Notes applications), and that will be deployed in an organization, administrators need to make sure that all signers associated with this code are trusted. You have these options:

- Add the signatures provided by the software vendor to your list of trusted signatures on your Administration ECL.
- Sign all new databases with an approved internal ID, using the Admin Tools - Sign utility for signing databases (see the topic "Signing a Template or Database" in **Domino 5 Administration Help**). This utility

operates on production databases; it takes a database template and signs all the design elements with a new signature.

For active content that is created internally, we offer the following approaches:

- Create special signing IDs, which exist for the sole intent of signing databases, templates, and code for ECL purposes, and give the IDs rights to run restricted agents and be included in Administration ECL. The IDs exist apart from admin IDs, and usage should be limited to those individuals authorized to sign content.

  In this scenario, it is extremely important to control access to the signing IDs. When authorized individuals leave the organization, their signing ID should be disabled. Similarly, new individuals who are given signing authority would get a new signing ID.
- Have a separate organizational unit within a organization for users who must sign templates and applications, and then create an ID in that organizational unit for each of those users (for example */Acme Template Developers/Acme). Users who create templates and applications should only use the IDs issued through the new organizational unit when signing their templates and applications. The Administration ECL can then be configured to trust any user in that special organizational unit.

Note: You should avoid wildcarding on trusted signatures (such as */JoesCompany) for an entire organization. Wildcarding in this instance means that all users within that organization are trusted. This is not recommended, primarily because most users don't, or don't need to, create active content; moreover, having such a policy in place makes any stolen ID potentially harmful.

See the **ECL access option risk levels sidebar** for two examples of ECL workstation security settings that show the levels of risk associated with each action for two signature policy scenarios - one for a very stringent signature policy (virtually no ESAs), and one for a less conservative policy.

Conclusion

ECLs are only effective if they are implemented properly. While the changes in 5.0.2 serve as gentle reminders about the presence and purpose of ECLs, it is up to Domino administrators to manage them effectively. This involves careful planning for who and what is trusted; thorough implementation of updated client ECLs; and ongoing maintenance of the Administration ECL, to reflect changes in trusted signers.

ABOUT AMY
Amy Smith is a principal user assistance writer for Lotus. She writes and maintains functional specs for Domino and Notes. She also is a member of the Notes UA Web team. Amy became interested in ECLs after getting one too many Execution Security Alerts. She practices good ECL hygiene and *never* trusts unsigned active content.

ABOUT CHARLIE
Charlie Kaufman is a security architect for Notes and Domino. He got his start in security breaking into systems using Trojan horses, but has spent the last 25 years on the other side trying to keep the bad guys out. He sees protecting users who don't care about security from active content as his toughest (and most fun!) challenge ever.

ABOUT CHUCK
Chuck Bassett is a software engineer working on Lotus Notes security at Iris. When he's not writing code he enjoys steep days in deep powder and long rides in the trees.

ABOUT MARY ELLEN
Mary Ellen Zurko is a security architect at Iris. For 13 years she has been working on putting security and usability together. She sees ECLs as an exciting challenge in this area.

SPECIAL THANKS

Special thanks to Katherine Spanbauer, of Lotus Professional Services, for her help with this article.

What do you think about this article?

Register Here!

About this Site | Feedback
Lotus Home | IBM Home | Iris Home

Powered by Iris Associates, Inc.

Home Download Iris Today Iris Cafe All About Domino Iris Sandbox Doc Library

[back to "Staying alert with Execution Control Lists "]

ECL access option risk levels

There are tradeoffs between user convenience (fewer execution security alerts) and tighter security. The tables below categorize the level of risk associated with each workstation security action for two signature policy scenarios.

Very stringent signature policy
Here is an example of a conservative ECL policy, which ensures fairly strong security with higher likelihood of ECL alerts.

Action	Risk	Default	No Signature	Lotus Notes Template Development/Lotus Notes	*/Organization	*/OU/Organization (where * corresponds to trusted users)
Access to the file system	High	Do not allow	Do not allow	Allow	Do not allow	Do not allow
Access to the current database*	High	Do not allow	Do not allow	Allow	Do not allow	Allow
Access to environment variables	Low	Do not allow	Do not allow	Allow	Do not allow	Allow
Access to non-Notes data	Medium	Do not allow	Do not allow	Allow	Do not allow	Allow
Access to external code (such as Notes LSX or API programs)	High	Do not allow	Do not allow	Allow	Do not allow	Do not allow
Access to external programs (such as non-Notes programs)	High	Do not allow	Do not allow	Allow	Do not allow	Do not allow
Ability to send mail	High	Do not allow	Do not allow	Allow	Do not allow	Allow
Ability to read other databases	Medium	Do not allow	Do not allow	Allow	Do not allow	Allow

Ability to modify other databases	Medium -High	Do not allow	Do not allow	Allow	Do not allow	Allow
Ability to export data	Medium	Do not allow	Do not allow	Allow	Do not allow	Allow
Access to Workstation Security ECL	High	Do not allow	Do not allow	Allow	Do not allow	Allow

*Access to current database includes both read and write access. This can be risky in the context of a user's mail file. Use caution when assigning this privilege to users. However, if a consistent signing policy does not exist, not allowing access to current database will generate a large number of Execution Security Alerts.

Less conservative signature policy
Here is an example of ECL that minimizes execution control alerts while mitigating only the most severe risks.

Action	Risk	Default	No Signature	Lotus Notes Template Development/ Lotus Notes	*/Organization	*/OU/Organization (where OU corresponds to trusted users)
Access to the file system	High	Do not allow	Do not allow	Allow	Do not allow	Do not allow
Access to the current database*	High	Do not allow	Do not allow	Allow	Allow*	Allow*
Access to environment variables	Low	Allow	Do not allow	Allow	Allow	Allow
Access to non-Notes data	Medium	Do not allow	Do not allow	Allow	Allow	Allow
Access to external code (such as Notes LSX or API programs)	High	Do not allow	Do not allow	Allow	Do not allow	Allow
Access to external programs (such as non-Notes programs)	High	Do not allow	Do not allow	Allow	Do not allow	Allow
Ability to send mail	High	Do not allow	Do not allow	Allow	Do not allow	Allow
Ability to read other databases	Medium	Allow	Do not allow	Allow	Allow	Allow
Ability to modify other	Medium-High	Do not allow	Do not allow	Allow	Do not allow	Allow

databases						
Ability to export data	Medium	Do not allow	Do not allow	Allow	Do not allow	Allow
Access to Workstation Security ECL	High	Do not allow	Do not allow	Allow	Do not allow	Allow

*Access to current database includes both read and write access. This can be risky in the context of a user's mail file. Use caution when assigning this privilege to users. However, if a consistent signing policy does not exist, not allowing access to current database will generate an increased number of Execution Security Alerts.

Notes.net Iris Today

Home Download Iris Today Iris Cafe All About Domino Iris Sandbox Doc Library

[back to "Staying alert with Execution Control Lists "]

Workstation access options

Choose from these options when setting up a workstation ECL:

Access option	**Allows formulas and code to**
Access to the file system	Attach, detach, read to, and write from workstation files
Access to current database	Read and modify the current database
Access to environment variables	Use the @SetEnvironment and @GetEnvironment variables and LotusScript methods to access the NOTES.INI file
Access to non-Notes databases	Use @DBLookup, @DBColumn, and @DBCommand to access databases when the first parameter for these @functions is a database driver of another application
Access to external code	Run LotusScript classes and DLLs that are unknown to Notes
Access to external programs	Access other applications, including activating any OLE object
Ability to send mail	Use functions such as @MailSend to send mail
Ability to read other databases	Read information in databases other than the current database
Ability to modify other databases	Modify information in databases other than the current database
Ability to export data	Print, copy to the clipboard, import, and export data
Access to Workstation Security ECL	Modify the ECL

Java applet options
Note: Although this article concentrates specifically on workstation security ECLs, descriptions of Java and JavaScript security ECL options are also provided here.

When a Java applet runs within Notes, certain security restrictions are imposed on that applet. This is sometimes referred to as the "Java security sandbox". This security model protects against malicious code by determining what operations an applet can perform and what system resources it can access. These restrictions can be customized on a per-signature basis by enabling the checkboxes as described below.

Access option	**Allows the applet to**
Access to file system	Read and write files on the local file system.
Access to Notes Java classes	Load and call the Domino back-end object classes.
Access to network addresses	Bind to and accept connections on a privileged port (a port outside the range 0 to 1024) and establish connections with other servers.
Printing	Submit print jobs.

Access to system properties	Read system properties such as color settings and environment variables.
Dialog and clipboard access	Access to the system Clipboard and also determines whether the "security banner" is displayed in top-level windows. The security banner is a visual indication (usually a message like "Java Applet Window") that this window was created by a Java applet. This is done to ensure that a user does not inadvertently enter security-sensitive information into a dialog masquerading as a password dialog, for example. Enabling this checkbox causes the security banner not to be displayed.
Process-level access	Create threads and threadgroups, fork and execute external processes, load and link external libraries, access non-public members of classes using Java core reflection, and access the AWT event queue.

JavaScript options
The JavaScript ECL options control security for JavaScript executing within the Notes client, either on a Notes form or on a Web page rendered by the Notes browser. These options do not control JavaScript executed by other browsers including the Microsoft Internet Explorer browser, even when embedded within the Notes client. The read and write options (under the general categories "Allow Read Data Access From" and "Allow Write Data Access To," respectively) control whether JavaScript code can read or modify JavaScript properties of the Window object. The Window object is the top-level object in the JavaScript document object model. It has properties that apply to the entire window. Securing access to the Window object secures access to other objects on the page since the JavaScript program cannot access the objects lower in the object model without first traversing the Window object.

You can control the security for these read and write options independently for three different classes of Window objects:

Window object class	Description
Source window	Controls JavaScript access to the Window object on the same page as the JavaScript code. Typically this is a very low security threat. Selecting this option does not prevent JavaScript calls if the call is made directly to the object on the source window. Doing so circumvents the Window object; therefore this ECL option is not enforced. The default is to allow read and write access.
Other window from same host	Controls JavaScript access to the Window object on a different page from the JavaScript code, but from a page using the same host. For example, JavaScript code on a page on www.lotus.com can access the Window object on another page on www.lotus.com. This allows two pages to interact if they are within the same frameset. This is a slightly higher security threat. The default is to allow read and write access.
Other window from different host	This is similar to "Other window from same host," except it enables access to the Window object on a different page within a frameset that uses a different host. For example, JavaScript code on a page on www.lotus.com can access the Window object on a page on any other server. This is the highest security threat because of the possibility of someone designing a frameset containing a page performing malicious actions accessing data on another page in the same frameset that you "trust," where you might type a password or some other sensitive information. The default is to not allow read and write access.

There are two additional ECL options that control whether JavaScript executing in the Notes client is authorized to open a new Web page or Notes document.

The following options are available in the "Allow Open Access To" category:

Option	**Description**
URL on same host	Controls access for opening a page or Notes document on the same host as the JavaScript code. The default is to allow open access.
URL on different host	Controls access for opening a page or Notes document on a different host as the JavaScript code. The default is to not allow open access.

Register Here!

About this Site | **Feedback**
Lotus Home | **IBM Home** | **Iris Home**

POWERED BY iris ASSOCIATES, INC.