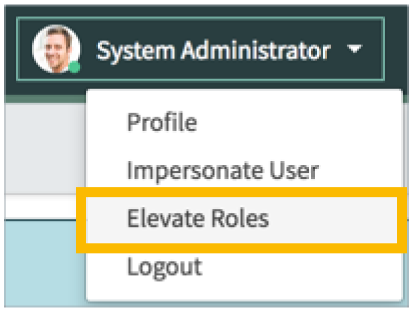
ServiceNow Application Developer

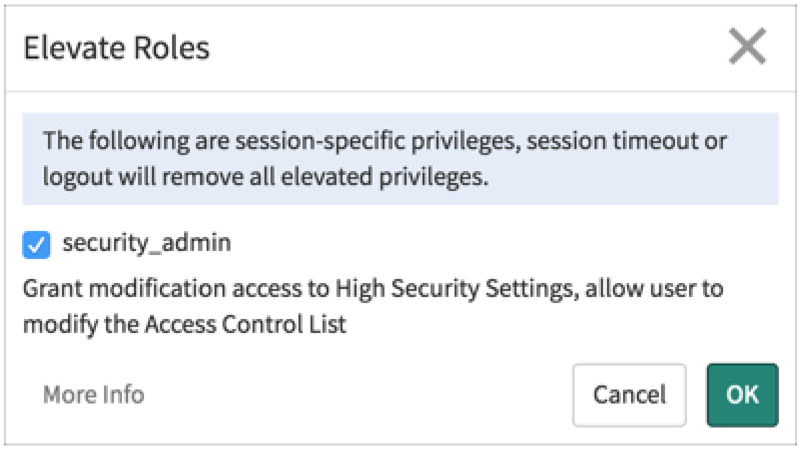
Securing Applications Against Unauthorized Users > Creating and Editing Access Controls

Users with the *admin* role can view Access Control records but cannot create, delete, or update Access Controls. All Access control operations except read require elevated security privileges. Elevated security privileges are also known as the *security_admin* role.

Elevate security privileges in the main ServiceNow browser window (not Studio). Open the **User menu** in the banner and select the **Elevate Roles** menu item.



In the *Elevate Roles* dialog, select *security_admin* then click the **OK** button.


Elevate Roles

The following are session-specific privileges, session timeout or logout will remove all elevated privileges.

☑ security_admin

Grant modification access to High Security Settings, allow user to modify the Access Control List

The *security_admin* role times out. When creating or editing Access Controls, it may be necessary to elevate roles more than once due to timing out.

Access Controls can be created for:

- Records
- Client callable Script Includes
- Processors
- REST Endpoints
- UI Pages

This module discusses Access Controls for records.

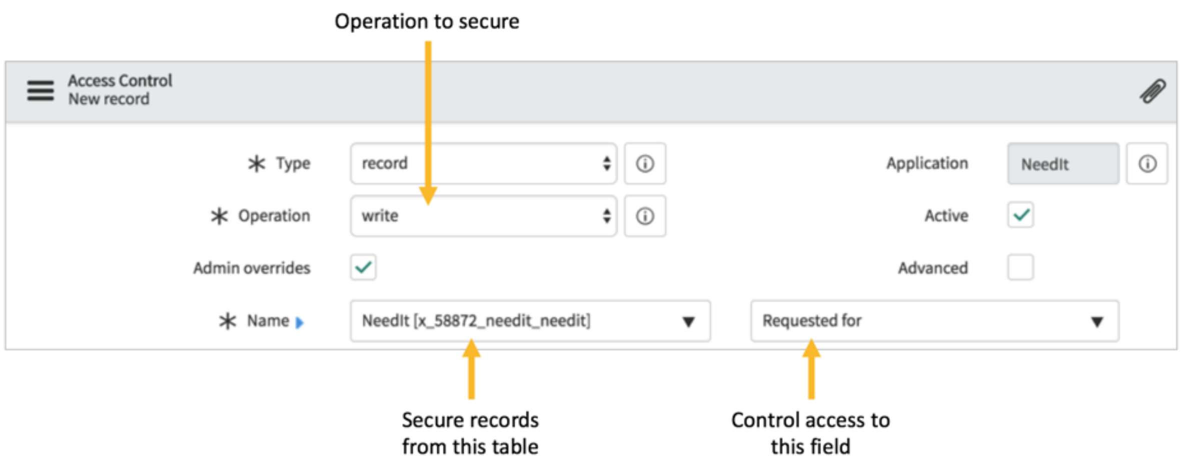
There are four sections in Access Controls:

- ***Table/field***
- ***Requires Role***
- ***Condition***
- ***Script***

In order for permission to be granted to access a table/field, the *Requires Role*, *Condition*, and *Script* sections must all return true. If there is no value, the section returns true.

Table/Field

Use the *Name* field in the Access Control configuration to specify which records and which field are secured. In the example, the Access Control grants write access to the *NeedIt* table's *Requested for* field.



Access Control
New record

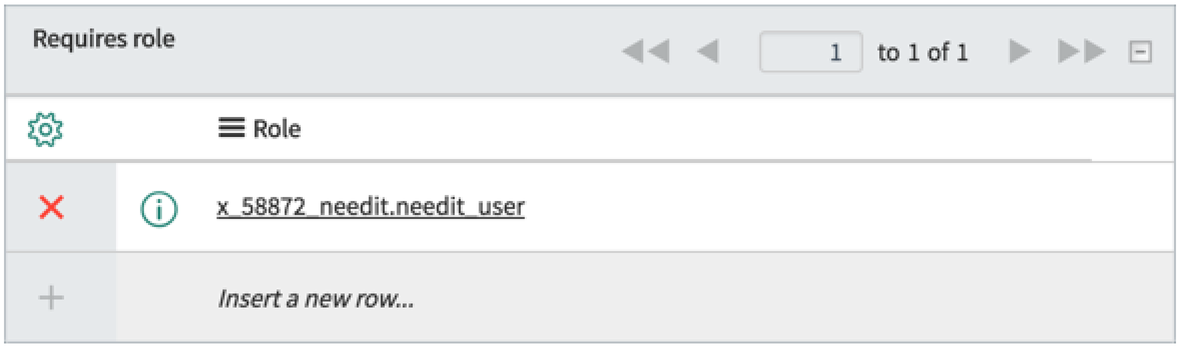
Type	record	Application	NeedIt
Operation	write	Active	✓
Admin overrides	✓	Advanced	
Name	NeedIt [x_58872_needit_needit]	Requested for	

The example rule is a write rule:

Write: [NeedIt] [Requested for]

Requires Role

Use the *Requires role* list to specify the role(s) required to access records. Click the **Insert a new row...** line to add a role to the list. If there are multiple rows in the list, the user only needs one of the roles for *Requires role* to return true.



Requires role

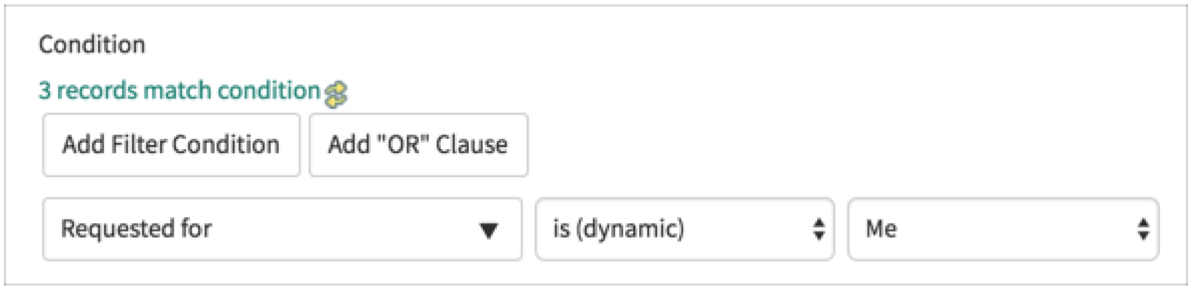
1 to 1 of 1

Role
x_58872_needit.needit_user
Insert a new row...

To remove a role from the list, click the **X** in the role's row. Clicking the **X** removes the role from the Access Control but does not delete the role from the database.

Condition

Use the *Condition* field to create the condition(s) required to grant access. In the example, the *Requested for* value must be the currently logged in user.


Condition

3 records match condition

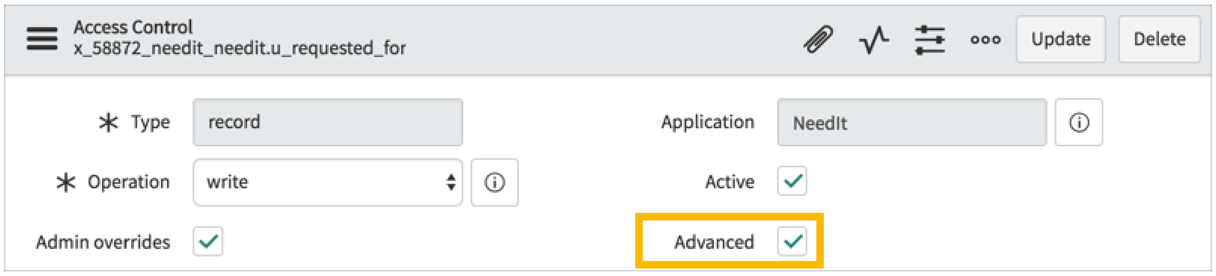
Add Filter Condition | Add "OR" Clause

Requested for | is (dynamic) | Me

The condition is tested dynamically and the number of matching records in the database is reported. Click the link to open a list of matching records in a new tab. Click the **Update count** icon ( ) to refresh the count.

Script

Select the *Advanced* option to see the *Script* field.



Access Control
x_58872_needit_needit.u_requested_for

Type	record	Application	NeedIt
Operation	write	Active	✓
Admin overrides	✓	Advanced	✓

Access Control scripts execute server-side. For best performance, avoid Access Control scripts that use GlideRecord queries as they can adversely impact performance.

Restrict the script logic only to security-related logic. If other logic is included, such as managing date formats or validating record data, it can be difficult to debug problems in the future as you might not think to look in Access Control scripts for those actions.

Some useful methods for Access Control scripts include:

- ***GlideSystem***: *getUser(), getUserID(), getUserName(), hasRole(), isLoggedIn(), isInteractive(), getSession()*
- ***GlideRecord***: *isNewRecord()*

Access Control scripts must set the *answer* variable to *true* or *false*.

```
if (!gs.hasRole("admin") && !gs.hasRole("user_admin") &&
gs.getSession().isInteractive()) {
    answer = true;
}
else{
    answer = false;
}
```