Security

The security metadata for an endpoint lets you describe a set of requirements that must be passed in order for the business logic to execute. Requirements are executed in collections. If any requirement in the collection passes the entire collection passes. A collection of requirements can be defined at the following locations:

- At the endpoint itself. By default, this group always has a requirement that states authentication is required.
    - Note: Since all requirements in a collection are processed with a 'one pass all pass' approach, the only security requirement you should add at the end point level is 'allowAnonymous'. Other requirements at this level will not have the intended effect. This is because of the implicit security requirement requiring authentication that's applied at this level. More information is available on the page for the Allow Anonymous Requirement.
- Within each route segment. In this scenario, while the requirements are defined at the segment level, the entire collection of route level requirements are treated as a single collection. Requirements picked up from parent routes are evaluated as separate collections.
- Within each input entity field definition.

Collections short circuit for failures. If the end point collection fails, the route and input entity collections will not execute. Requirements within a collection short circuit on success, since one requirement passing means the entire collection passes.

Most security requirements are implemented as database functions but there are a few code based requirements too. A security block is a JSON object whose properties are a descriptive name for the requirement. The property name will be used in error messages if the requirement fails to pass. The property values are JSON objects that describe additional metadata needed to configure or execute the requirement. The structure of these property values will vary across requirements of different types. The only common property that all requirements share is the 'type' property, which gives the framework enough information to instantiate the objects necessary to execute the requirement.

Common Metadata for Security Requirements

Property Name	Value	Description
type	string	The type of the requirement in question. This tells the framework how to instantiate the objects necessary to execute the requirement.

The security framework is flexible enough to allow for requirements of any type to be created but that is not covered in these pages.

Additional Considerations

Be very aware of the data you are exposing through services. Your services site is **always** going to be accessible regardless of how you are securing your front end pages. **Putting your pages behind a CMS does not protect you.** It is trivial for users to open the network tab on the browser, see what requests are being made, and make manual calls to mine for additional data. Let's assume you want to expose an endpoint to retrieve information from persons. Consider the following questions:

- Is your requirement that the current logged in person can only view their own information and never view information of others? If it is, **do not** accept a person ID as input from the client. There's no need to expose yourself to the risk if it is not necessary.
- Is your requirement that the current logged in person can view personal information of other users? If it is, you're going to need to accept the person id as input from the client and **you must add a requirement that restricts the allowed persons to the smallest list possible.** This could be done through a SQL Function requirement or a custom code based requirement.