

# Permission Resource

*Check APIs and Client Usage*

Version 1.2.2-SNAPSHOT, 2025-11-08T20:59:18Z

# Table of Contents

1. APIs .....	2
1.1. POST /permission/check .....	2
1.2. POST /permission/check-with-index .....	2
1.3. Response example (legacy rules present and requiresServer=true) .....	4
2. Data-Domain Scope and Fallback .....	8
3. JavaScript Client Library .....	9
3.1. Using in a Browser (no build tools) .....	9
3.2. Using in React .....	10
3.3. Using in Vue .....	10
4. cURL Examples .....	12
5. Caching Guidance .....	13
6. Troubleshooting .....	14

This guide explains how to use the Permission Resource check APIs and how to evaluate the server-produced access decisions on the client using the provided JavaScript library.

It covers:

- The `/check` API (server-evaluated permission)
- The `/check-with-index` API (client-evaluatable snapshot)
- Request payloads and detailed response structures
- Scope and data-domain fallback behavior
- Using the JavaScript client in a browser, React, or Vue



This document reflects the current and evolving server behavior. The JavaScript client library is available at runtime from Quarkus as `/security/acl-client.js` and is forward-compatible with the scoped access-matrix format described here.

# Chapter 1. APIs

## 1.1. POST /permission/check

Performs a server-side permission check for a single request using the caller identity and an optional data domain (organization/account/tenant/segment/owner).

*Request*

```
{  
    "identity": "user-123",  
    "realm": "b2bi",  
    "area": "security",  
    "functionalDomain": "userProfile",  
    "action": "view",  
    "resourceId": "12345",  
    "orgRefName": "acme",  
    "accountNumber": "A1",  
    "tenantId": "t-001",  
    "dataSegment": 0,  
    "ownerId": "user-123",  
    "roles": ["user", "admin"],  
    "scope": "api"  
}
```

*Response (example)*

```
{  
    "finalEffect": "ALLOW",  
    "winningRule": "SysAnyActionSecurity",  
    "explanations": [  
        { "rule": "SysAnyActionSecurity", "effect": "ALLOW" }  
    ]  
}
```

Notes: - This endpoint is authoritative and understands scripts/postconditions and any dynamic runtime evaluation. - Use `/check` when you must account for scripts, context-enriched rules, or when you don't have the precomputed snapshot.

## 1.2. POST /permission/check-with-index

Returns a precomputed permission snapshot for the supplied identity. This snapshot can be cached on the client and used for fast allow/deny decisions without server roundtrips.

Depending on server version/config, the response can include:

- A legacy flat list of rules (for backward compatibility)
- A scoped access matrix (recommended) keyed by data-domain scope → area → domain → action

*Request (current servers; send data-domain as top-level fields)*

```
{  
  "identity": "user-123",  
  "realm": "b2bi",  
  "orgRefName": "acme",  
  "accountNumber": "A1",  
  "tenantId": "t-001",  
  "dataSegment": 0,  
  "ownerId": "user-123"  
}
```



If your server version supports a nested dataDomain object, you may also send:

```
{  
  "identity": "user-123",  
  "realm": "b2bi",  
  "dataDomain": {  
    "orgRefName": "acme",  
    "accountNumber": "A1",  
    "tenantId": "t-001",  
    "dataSegment": 0,  
    "ownerId": "user-123"  
  }  
}
```

On servers that do not support nested dataDomain, this shape will produce: **400 Bad Request: Unrecognized field "dataDomain".**

*Response: Scoped access matrix (recommended shape)*

```
{  
  "enabled": true,  
  "version": 42,  
  "policyVersion": 123,  
  "sources": ["user:user-123", "role:user", "role:admin"],  
  "requiresServer": false,  
  
  "scopes": {  
    "org=acme|acct=A1|tenant=t-001|seg=0|owner=user-123": {  
      "requiresServer": false,  
      "matrix": {  
        "security": {  
          "userProfile": {  
            "view": { "effect": "ALLOW", "rule": "ViewOwnProfile", "priority": 5,  
"finalRule": true, "source": "role:user" }  
          },  
          "credential": {  
            "update": { "effect": "DENY", "rule": "NoUpdate", "priority": 10,  
"finalRule": true, "source": "role:user" }  
          }  
        }  
      }  
    }  
  }  
}
```

```

    "finalRule": true, "source": "role:user" }
        }
    },
    "orders": {
        "manage": { "*": { "effect": "DENY", "rule": "NoManage", "priority": 50,
"finalRule": true } }
    }
},
{
    "org=acme|acct=A1|tenant=t-001|seg=*|owner=*": { "requiresServer": false, "matrix"
": { "*": { "*": { "*": { "effect": "DENY", "rule": "DefaultDeny", "priority": 999,
"finalRule": false } } } } },
    "org=*|acct=*|tenant=*|seg=*|owner=*": { "requiresServer": false, "matrix": {
"security": { "*": { "*": { "effect": "ALLOW", "rule": "SysRoleAnyActionSecurity",
"priority": 1, "finalRule": true } } } }
},
    "requestedScope": "org=acme|acct=A1|tenant=t-001|seg=0|owner=user-123",
"requestedFallback": [
    "org=acme|acct=A1|tenant=t-001|seg=0|owner=*",
    "org=acme|acct=A1|tenant=t-001|seg=*|owner=*",
    "org=acme|acct=A1|tenant=*|seg=*|owner=*",
    "org=acme|acct=*|tenant=*|seg=*|owner=*",
    "org=*|acct=*|tenant=*|seg=*|owner=*"
],
"rules": [
    { "name": "SysRoleAnyActionSecurity", "uri": "system:security:*:*:*:*:*:*",
"effect": "ALLOW", "priority": 1, "finalRule": true }
]
}
}

```

Interpretation: - The client should prefer the scope that best matches its current data domain and then look up `area → domain → action` in that scope's matrix, falling back through `requestedFallback`. - Within a matrix, exact values beat wildcards; the matrix already encodes the winning outcome per triple. - If `requiresServer` is true (globally or for a specific scope), the client should call `/check` for decisions in those affected areas.

## 1.3. Response example (legacy rules present and `requiresServer=true`)

```
{
    "enabled": false,
    "version": 0,
    "policyVersion": 163044986023000,
    "rules": [
        {
            "name": "users can't delete anything in security area",
            "uri": "user:security*:delete|*:*:*:*:*:*",

```

```

    "effect": "DENY",
    "priority": 10,
    "finalRule": true
},
{
  "name": "view your own resources",
  "uri": "user:*:*:system-com:*:*:*:*:*",
  "effect": "ALLOW",
  "priority": 10,
  "finalRule": true
},
{
  "name": "view your own resources, limit to default dataSegment",
  "uri": "user:*:*|*:*:system@system.com:*",
  "effect": "ALLOW",
  "priority": 10,
  "finalRule": false
},
{
  "name": "ViewSystemResources",
  "uri": "user:*:view|system-com:*:*:system@system.com:*",
  "effect": "ALLOW",
  "priority": 10,
  "finalRule": true
}
],
"sources": [
  "user"
],
"requiresServer": true,
"scopes": {
  "org=*|acct=*|tenant=*|seg=*|owner=system@system.com": {
    "matrix": {
      "*": {
        "*": {
          "view": {
            "effect": "ALLOW",
            "rule": "ViewSystemResources",
            "priority": 10,
            "finalRule": true,
            "source": "user"
          }
        }
      }
    }
  },
  "requiresServer": false
},
"org=*|acct=*|tenant=*|seg=*|owner=*": {
  "matrix": {
    "security": {
      "*": {

```

```

        "delete": {
            "effect": "DENY",
            "rule": "users can't delete anything in security area",
            "priority": 10,
            "finalRule": true,
            "source": "user"
        }
    }
},
"*": {
    "*": {
        "*": {
            "effect": "ALLOW",
            "rule": "view your own resources, limit to default dataSegment",
            "priority": 10,
            "finalRule": false,
            "source": "user"
        }
    }
},
"requiresServer": false
}
},
"requestedScope": "org=acme|acct=A1|tenant=t-001|seg=0|owner=user-123",
"requestedFallback": [
    "org=acme|acct=A1|tenant=t-001|seg=0|owner=*",
    "org=acme|acct=A1|tenant=t-001|seg=*|owner=*",
    "org=acme|acct=A1|tenant=*|seg=*|owner=*",
    "org=acme|acct=*|tenant=*|seg=*|owner=*",
    "org=*|acct=*|tenant=*|seg=*|owner=*"
]
}
}

```

## Field semantics

- enabled: false indicates the compiled index is disabled or unavailable. Clients should treat the snapshot as non-authoritative and prefer calling /permission/check for critical decisions; the matrix may still be present for some scopes but is not guaranteed complete.
- version: 0 accompanies enabled=false. When enabled is true, version corresponds to the compiled index version and can be used for caching together with policyVersion.
- policyVersion: the ruleset/policy timestamp or version for cache invalidation.
- rules: legacy flat list preserved for backward compatibility. Clients should prefer the scoped matrix when available.
- sources: identities included when the snapshot was materialized (e.g., user id and/or roles).
- requiresServer (top-level): true means at least one rule could not be safely materialized (e.g., uses scripts/postconditions or dynamic filters). Clients should be prepared to call /permission/check for affected scopes/decisions.

- `scopes[<key>].requiresServer`: per-scope flag. If true, client should not rely on that scope's matrix for final decisions and should call `/permission/check` when evaluating in that scope.
- `requestedScope` / `requestedFallback`: convenience keys provided when the request included data-domain values. Clients should attempt lookup starting at `requestedScope`, then walk `requestedFallback` in order.

## Chapter 2. Data-Domain Scope and Fallback

A scope key is a canonical string combining the data-domain dimensions:

```
org=<v>|acct=<v>|tenant=<v>|seg=<v>|owner=<v>
```

- Values are specific strings or `*` for wildcard.
- Fallback traversal order: owner → segment → tenant → account → org → global.

# Chapter 3. JavaScript Client Library

The JavaScript client provides helpers to evaluate the snapshot on the client.

- Served by Quarkus at: [/security/acl-client.js](#)
- Path in repo: [quantum-framework/src/main/resources/META-INF/resources/security/acl-client.js](#)

*Exported API*

```
ACLClient.scopeKeyFromDataDomain(dataDomain) // => scope key string
ACLClient.buildFallbackChain(scopeKey)        // => [less-specific scope keys]
ACLClient.lookupAreaDomainAction(matrix, area, domain, action) // => Outcome | null
ACLClient.decide(snapshot, dataDomain, area, domain, action)   // => 'ALLOW' | 'DENY'
ACLClient.decideOutcome(snapshot, dataDomain, area, domain, action) // => Outcome | null
```

Outcome structure:

```
{
  "effect": "ALLOW",
  "rule": "<winning rule name>",
  "priority": 0,
  "finalRule": true,
  "source": "role:user"
}
```

Allowed effect values are "ALLOW" or "DENY".

## 3.1. Using in a Browser (no build tools)

*HTML*

```
<script src="/security/acl-client.js"></script>
<script>
  async function canViewProfile() {
    // 1) Get a snapshot for the user
    const res = await fetch('/permission/check-with-index', {
      method: 'POST', headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ identity: 'user-123', realm: 'b2bi', orgRefName: 'acme',
accountNumber: 'A1', tenantId: 't-001', dataSegment: 0, ownerId: 'user-123' })
    });
    const snapshot = await res.json();

    // 2) Evaluate locally
    const decision = ACLClient.decide(snapshot, { orgRefName: 'acme', accountNumber:
'A1', tenantId: 't-001', dataSegment: 0, ownerId: 'user-123' }, 'security',
'userProfile', 'view');
    if (decision === 'ALLOW') {
```

```

    // show UI
} else {
    // hide or show alternative
}
}

</script>

```

## 3.2. Using in React

*Installation (served by your Quarkus backend)*

- No npm package is required; include as an external script in `public/index.html` or via dynamic import.

*React example*

```

import { useEffect, useState } from 'react';

export default function ProfileButton() {
    const [allowed, setAllowed] = useState(false);

    useEffect(() => {
        async function run() {
            const res = await fetch('/permission/check-with-index', {
                method: 'POST', headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify({ identity: 'user-123', realm: 'b2bi' })
            });
            const snapshot = await res.json();
            const effect = window.ACClient.decide(snapshot, null, 'security',
                'userProfile', 'view');
            setAllowed(effect === 'ALLOW');
        }
        run();
    }, []);
}

if (!allowed) return null;
return <button>View Profile</button>;
}

```

## 3.3. Using in Vue

*Vue component example*

```

<template>
    <button v-if="allowed">View Profile</button>
</template>

<script>
export default {

```

```
data() { return { allowed: false }; },
async mounted() {
  const res = await fetch('/permission/check-with-index', {
    method: 'POST', headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ identity: 'user-123', realm: 'b2bi' })
  });
  const snapshot = await res.json();
  const effect = window.ACClient.decide(snapshot, null, 'security', 'userProfile',
  'view');
  this.allowed = (effect === 'ALLOW');
}
</script>
```

# Chapter 4. cURL Examples

*Server-evaluated check*

```
curl -sS -X POST \
-H 'Content-Type: application/json' \
http://localhost:8080/permission/check \
-d '{
  "identity": "user-123",
  "realm": "b2bi",
  "area": "security",
  "functionalDomain": "userProfile",
  "action": "view",
  "orgRefName": "acme",
  "accountNumber": "A1",
  "tenantId": "t-001",
  "dataSegment": 0,
  "ownerId": "user-123"
}'
```

*Client-evaluatable snapshot*

```
curl -sS -X POST \
-H 'Content-Type: application/json' \
http://localhost:8080/permission/check-with-index \
-d '{
  "identity": "user-123",
  "realm": "b2bi",
  "orgRefName": "acme",
  "accountNumber": "A1",
  "tenantId": "t-001",
  "dataSegment": 0,
  "ownerId": "user-123"
}'
```

# Chapter 5. Caching Guidance

- Clients should cache the `/check-with-index` response keyed by (`identity`, `realm`, `version`, `policyVersion`).
- Refresh the snapshot when either `version` or `policyVersion` changes.

# Chapter 6. Troubleshooting

- If `requiresServer` is `true` (globally or per-scope), call `/check` for affected decisions.
- If no matrix entry is found in any scope fallback, default to `DENY` for safety.
- Ensure `effect` comparisons are case-insensitive on the client (`String(effect).toUpperCase()`).