OAuth Scopes and Permissions

Overview

OAuth scopes provide granular, fine-grained permission control within Genesys Cloud. They define exactly what resources an application can access and what actions it can perform. Scopes are requested during authorization and enforced on every API call.

Scope Fundamentals

```
What Are Scopes?

Definition:
├─ Granular permissions for API access
├─ Limit what an application can do
├─ User consent on what app can access
├─ Enforced on every API request
└─ Principle of least privilege

Scope Format:
  resource:action
  OR
  resource:action:scope

Examples:
├─ conversations:readonly       # View conversations
├─ conversations:call:add       # Make calls
├─ conversations:call:control   # Transfer, hold
├─ users:manage                 # Create/modify users
```

```
├─ routing:queue:view          # View queues
└─ scheduling:manage           # Manage schedules

Space-Separated Combination:
  "conversations:readonly users:readonly scheduling:manage"

User Sees During Authorization:
├─ App name
├─ All requested scopes
├─ What the app can do
└─ User grants all or none (all-or-nothing)

Enforcement:
├─ Every API request validated against token scopes
├─ If endpoint requires scope not in token: 403 Forbidden
├─ User permissions AND scopes both required (AND logic)
├─ Whichever is more restrictive applies
└─ Better to have both than just one
```

Scope Categories

```
Conversation Management:
├─ conversations:readonly       # GET conversations
├─ conversations:call:add       # POST/make calls
├─ conversations:call:control   # Transfer, hold, disconnect
├─ conversations:call:pull      # Pull calls
├─ conversations:call:coach     # Coach calls
├─ recordings:view              # Access recordings
├─ recordings:download          # Download recording files
├─ transcripts:readonly         # View transcripts
└─ conversations:external:contact:add # Add external contacts

User Management:
├─ users:readonly               # Read user info
├─ users:manage                 # Create/update/delete users
├─ authorization:grant:readonly # View user permissions
├─ authorization:grant:manage   # Modify user permissions
```

```
├─ presence:readonly            # View presence status
└─ presence:manage              # Change presence

Contact Center Configuration:
├─ routing:queue:view           # View queue config
├─ routing:queue:manage         # Create/edit queues
├─ routing:skill:view           # View skills
├─ routing:skill:manage         # Manage skills
├─ directory:organization:view  # View org structure
├─ telephony:phone:manage       # Manage phones
├─ outbound:campaign:view       # View campaigns
└─ outbound:campaign:execute    # Run campaigns

Workforce Management:
├─ forecasting:readonly         # View forecasts
├─ forecasting:manage           # Create/edit forecasts
├─ scheduling:readonly          # View schedules
├─ scheduling:manage            # Create/edit schedules
├─ timeoff:view                 # View time-off requests
├─ timeoff:manage               # Approve time-off
├─ adherence:view               # View adherence
└─ adherence:manage             # Manage adherence

Analytics & Reporting:
├─ analytics:conversationDetail # Detailed conversation analytics
├─ analytics:aggregate:view     # Aggregated analytics
├─ quality:evaluation:view      # Quality evaluations
├─ quality:evaluation:manage    # Create/edit evaluations
├─ speechanalytics:data:view    # Speech analytics
└─ reporting:analytics:view     # Analytics reports

Integration & External:
├─ externalcontacts:manage      # CRM synchronization
├─ webhooks:manage              # Webhook configuration
├─ webhooks:view                # View webhooks
├─ scim:write                   # SCIM provisioning
├─ knowledge:manage             # Knowledge articles
└─ knowledge:readonly           # View knowledge

Platform Management:
```

```
├─ oauth:client:view          # View OAuth clients
├─ oauth:client:manage        # Create/edit OAuth clients
├─ admin:org:view             # View org settings
├─ admin:org:manage           # Modify org settings
└─ audit:readonly             # View audit logs
```

Scope Selection Best Practices

Principle of Least Privilege:

Definition:
- ├─ Grant minimum scopes needed
- ├─ Only request what app actually uses
- ├─ Reduce impact if app compromised
- ├─ Easier to review and audit
- └─ Better security posture

How to Select:

1. List API Endpoints Used:
   - ├─ /conversations → conversations:readonly
   - ├─ /v2/users/{id} → users:readonly
   - ├─ /v2/conversations/{id}/notes → conversations:readonly
   - └─ Map each endpoint to scope

2. Check Endpoint Documentation:
   - ├─ Visit API Explorer
   - ├─ View each endpoint
   - ├─ Note required scope
   - └─ Collect all scopes

3. Start with Read-Only:
   - ├─ conversations:readonly (not call:add)
   - ├─ users:readonly (not manage)
   - ├─ Only add write scopes if needed
   - └─ Safer starting point

4. Add Write Scopes Only If Needed:
   ├─ Review which operations require writes
   ├─ Add specific action scopes
   ├─ Example: call:add (not call:control)
   └─ One scope per action where possible

5. Remove Unused Scopes:
   ├─ Quarterly review
   ├─ Check OAuth client usage
   ├─ Remove unneeded scopes
   ├─ Reduce security surface
   └─ Document rationale

6. Document Your Choices:
   ├─ Why each scope is needed
   ├─ Which endpoints use it
   ├─ When to update scopes
   └─ Version control reasons

Example Selection:

App: Contact center agent desktop
├─ Endpoints needed:
│ ├─ GET /conversations → conversations:readonly
│ ├─ POST /conversations/{id}/participants/calls/transfer
│ │ → conversations:call:control
│ ├─ GET /users/me → users:readonly
│ └─ PATCH /v2/users/{id}/presence → presence:manage
│
└─ Final scopes: "conversations:readonly conversations:call:control users:readonly presence:manage"

NOT these (too permissive):
├─ conversations:manage (not needed)
├─ users:manage (not needed)
└─ conversations:call:add (can't initiate calls)

Scope Enforcement Mechanism

How Scopes Are Enforced:

Every API Request Validation:

1. Client Sends Request:
   GET /api/v2/users/123
   Authorization: Bearer {access_token}

2. Genesys Validates Token:
   ├─ Token signature valid?
   ├─ Token not expired?
   ├─ Token not revoked?
   └─ Check scopes and permissions

3. Endpoint Requires:
   ├─ Endpoint /v2/users/{id} requires scope: users:readonly
   ├─ Check if token has "users:readonly" scope
   └─ Check if user has permission to read users

4. Dual Validation (Both Required):
   ├─ Must have OAuth scope: ✓ users:readonly
   ├─ Must have user permission: ✓ (has role allowing read)
   ├─ Both? → Grant access
   ├─ Only one? → 403 Forbidden
   └─ Neither? → 403 Forbidden

5. Enforce Rules:
   ├─ User has scope but not permission → 403
   ├─ User has permission but not scope → 403
   ├─ User has both → 200 OK (success)
   └─ User has neither → 403

Example Scenarios:

Scenario 1: Has Scope, Has Permission
├─ Token scope: users:readonly ✓
├─ User role permission: can read users ✓
├─ Result: 200 OK (access granted)
└─ API call succeeds

Scenario 2: Has Scope, No Permission

├─ Token scope: users:readonly ✓

├─ User role permission: cannot read users ✗

├─ Result: 403 Forbidden

└─ API call denied

Scenario 3: No Scope, Has Permission

├─ Token scope: users:readonly ✗

├─ User role permission: can read users ✓

├─ Result: 403 Forbidden

└─ API call denied

Scenario 4: No Scope, No Permission

├─ Token scope: users:readonly ✗

├─ User role permission: cannot read users ✗

├─ Result: 403 Forbidden

└─ API call denied

Example Error Response:

HTTP 403 Forbidden

```
{
  "error": {
    "message": "This application is not authorized to perform this action",
    "code": "PERMISSIONS_INSUFFICIENT",
    "status": 403
  }
}
```

Checking Available Scopes:

In API Explorer:

1. Visit: https://developer.genesys.cloud/api/rest/v2/
2. Click any endpoint
3. See "Authorization" section
4. Lists required scope
5. Copy exact scope name

Common Scope Combinations

Agent Desktop Application:
  conversations:readonly
  conversations:call:control
  users:readonly
  presence:manage
  recordings:view
  (Read conversations, manage calls, see presence, view recordings)

Admin Dashboard:
  users:readonly
  routing:queue:view
  routing:skill:view
  scheduling:readonly
  analytics:conversationDetail
  (Read-only access to key admin features)

WFM Application:
  scheduling:manage
  forecasting:manage
  timeoff:manage
  adherence:manage
  users:readonly
  (Full WFM management)

Integration Service (Salesforce Sync):
  externalcontacts:manage
  users:readonly
  conversations:readonly
  (Sync contacts, read-only other data)

Chatbot / Virtual Agent:
  conversations:external:contact:add
  knowledge:readonly
  (Add external contacts, access knowledge)

Analytics Reporter:

analytics:conversationDetail
analytics:aggregate:view
quality:evaluation:view
speechanalytics:data:view
(Read-only analytics access)

API Automation (Least Privilege):
users:readonly (query users)
outbound:campaign:view (check campaign status)
(Minimal access for specific automation)

Scope Changes & Updates

Adding Scopes to Existing OAuth Client:

Scenario:
├─ App previously only read conversations
├─ Now needs to transfer calls
├─ Must add conversations:call:control scope

Steps:

1. Stop Using Old Client (or keep both briefly):
   ├─ Update app configuration
   ├─ Point to updated client
   └─ Test thoroughly

2. Navigate to OAuth Client in Admin:
   ├─ Admin → Integrations → OAuth
   ├─ Find your OAuth client
   ├─ Click Edit
   └─ Select Scopes

3. Update Scopes:
   ├─ Old: "conversations:readonly"
   ├─ New: "conversations:readonly conversations:call:control"
   ├─ Add all needed scopes

└ Save changes

4. User Must Re-Authorize:
├ For Authorization Code Grant: Yes
├ Users see new permissions consent screen
├ Users must grant new scope
└ New token includes updated scopes

5. Test Thoroughly:
├ Test new functionality
├ Verify old functions still work
├ Check error responses
└ Monitor for issues

6. Gradual Rollout (Optional):
├ Update small group first
├ Monitor for errors
├ Expand to more users
└ Full rollout when confident

Removing Unused Scopes:

Scenario:
├ App no longer uses contact creation
├ Can remove conversations:external:contact:add
├ Reduce security surface

Steps:

1. Verify Not Used:
├ Check code for usage
├ Search for endpoint calls
├ Verify in logs (not used)
└ Confirm removal safe

2. Update OAuth Client:
├ Admin → Integrations → OAuth
├ Click Edit
├ Deselect unused scope
└ Save

3. No Re-Authorization Needed:
   ├─ Existing tokens still work
   ├─ New tokens won't have old scope
   ├─ Endpoints still work (user has permission)
   └─ Gradual transition

4. Monitor for Errors:
   ├─ Watch logs for 403 errors
   ├─ Verify no broken functionality
   └─ Quick rollback if issues

Scope Testing

Testing Scope-Based Authorization:

Setup Test OAuth Client:

1. Create OAuth client with specific scopes
2. Authenticate with that client
3. Test that allowed endpoints work
4. Test that forbidden endpoints fail

Testing Allowed Scope:

Endpoint: GET /conversations
Scope: conversations:readonly
Token has scope: Yes

Test:
```
  curl -H "Authorization: Bearer {token}" \
       https://api.mypurecloud.com/api/v2/conversations
```

Expected: HTTP 200 (success)
Actual: ?

Testing Denied Scope (Negative Test):

Endpoint: POST /conversations (create)

Scope needed: conversations:manage

Token has scope: No

Test:

```
curl -X POST \
    -H "Authorization: Bearer {token}" \
    https://api.mypurecloud.com/api/v2/conversations
```

Expected: HTTP 403 Forbidden

Actual: ?

Test Multiple Scopes:

Token has scopes:

├─ conversations:readonly

├─ users:readonly

└─ NOT: scheduling:manage

Tests to Run:

├─ GET /conversations → 200 (has scope)

├─ GET /users → 200 (has scope)

├─ GET /schedules → 403 (no scope)

├─ DELETE /conversations/{id} → 403 (no manage scope)

└─ DELETE /users/{id} → 403 (no manage scope)

Automated Testing:

Example Test Suite:

Test Cases:

├─ [✓] conversations:readonly allows GET

├─ [✓] conversations:readonly denies POST

├─ [✓] conversations:call:control allows transfer

├─ [✓] users:readonly allows GET /users

├─ [✓] users:readonly denies PUT /users

├─ [✓] Multiple scopes work correctly

├─ [✓] Missing scopes return 403

└─ [✓] Expired token returns 401

Before Deployment:

├─ Run all scope tests

├─ Verify expected 403s

├─ Verify expected 200s

└─ Document scope requirements

Troubleshooting Scope Issues

Problem: Getting 403 Forbidden on Valid Endpoint

Check List:

┌─ Token Valid?

│ ├─ Is token expired?

│ ├─ Try making another call

│ └─ Get fresh token if needed

│

├─ Scope Correct?

│ ├─ Check API Explorer for required scope

│ ├─ Verify token has that scope

│ ├─ Look at token claims if JWT

│ └─ Add missing scope to OAuth client

│

├─ User Has Permission?

│ ├─ User role has permission?

│ ├─ Check user's roles in Admin UI

│ ├─ Check division assignments

│ └─ Add necessary role if missing

│

└─ Both Needed?

  ├─ User permission: Required

  ├─ OAuth scope: Required

  ├─ Have both? Should work

  └─ Missing one? 403 error

Example Troubleshooting:

Error: POST /v2/users returns 403
├─ Required scope: users:manage
├─ Check 1: Token has users:manage? NO ✗
│ └─ Solution: Add users:manage scope to OAuth client
│
├─ Check 2: User has create user permission? YES ✓
│ └─ OK (permission is good)
│
├─ Root Cause: Missing scope in token
├─ Solution: Update OAuth client scopes
└─ Retry after user re-authenticates

Error: GET /v2/conversations returns 403
├─ Required scope: conversations:readonly
├─ Check 1: Token has conversations:readonly? YES ✓
│ └─ OK (scope is good)
│
├─ Check 2: User has read conversations permission? NO ✗
│ └─ Solution: Add user to role with permission
│
├─ Root Cause: User lacks role permission
├─ Solution: Assign appropriate role to user
└─ Retry after role assignment takes effect

Common Mistakes:

 Wrong Scope Name:
├─ Requested: "conversation:view" (wrong)
├─ Correct: "conversations:readonly"
└─ Solution: Check API Explorer for exact scope

 Typo in Scope:
├─ Requested: "users :manage" (space)
├─ Correct: "users:manage"
└─ Solution: Check spacing carefully

 Missing Scope Entirely:
├─ Requested: "users:readonly" only
├─ Needed: "users:readonly users:manage"
└─ Solution: Add missing scopes

 Wrong Colon Format:
├─ Requested: "users-manage" (dash)
├─ Correct: "users:manage" (colon)
└─ Solution: Use colons, not dashes

Scope Documentation

For Your Team:

Document Your Scopes:

Application: [App Name]
├─ conversations:readonly
│ ├─ Used for: Display conversation history
│ ├─ Endpoints: GET /v2/conversations
│ └─ Rationale: Need to show past interactions
│
├─ conversations:call:control
│ ├─ Used for: Transfer calls between agents
│ ├─ Endpoints: POST /v2/conversations/{id}/participants/calls/transfer
│ └─ Rationale: Core feature for agent desktop
│
├─ users:readonly
│ ├─ Used for: List available agents
│ ├─ Endpoints: GET /v2/users
│ └─ Rationale: Show agent list for transfers
│
└─ Updated: March 2026
   └─ Next Review: September 2026

Scope Change Log:

Date	Change	Reason
2026-03-01	Added: call:control	New transfer feature
2026-01-15	Added: users:readonly	Show agent list

2025-11-20 | Initial: conversations:readonly | Read-only access

API Endpoint to Scope Mapping:

Endpoint	Method	Scope Required
/v2/conversations	GET	conversations:readonly
/v2/conversations	POST	conversations:readonly (new)
/v2/conversations/{id}	GET	conversations:readonly
/v2/users	GET	users:readonly
/v2/users/{id}/presence	PATCH	presence:manage

Key Takeaways: Chapter 5

- **Granular Permissions** - Scopes define exactly what app can do
- **Least Privilege** - Request minimum scopes needed
- **User Consent** - Users see all scopes during authorization
- **Dual Enforcement** - User permissions AND OAuth scopes both required
- **403 Means Missing** - Either scope or permission (or both) missing
- **Standard Format** - Use resource:action or resource:action:scope
- **Space-Separated** - Combine multiple scopes with spaces
- **Review Regularly** - Quarterly scope audits reduce security risk

---

Interview Prep: OAuth Scopes

Question	Answer
What are scopes?	Granular permissions defining what app can access/do
Scope format?	resource:action or resource:action:scope (e.g., conversations:readonly)
How combined?	Space-separated list (e.g., "conversations:readonly users:manage")
User sees scopes?	Yes, during authorization consent screen
All-or-nothing?	Yes, user grants all requested scopes or none
How enforced?	Every API request checked against token scopes
User + scope?	BOTH required (AND logic), not OR

Question	Answer
403 means?	Missing scope or missing permission (or both)
Best practice?	Principle of least privilege - request minimum needed
How update?	Edit OAuth client, add/remove scopes, user must re-auth

Document Version

Chapter: 5 of 8
Last Updated: March 2026
Status: Current with OAuth 2.0 standards
Scope: Scope management, enforcement, best practices

Revision #1
Created 14 March 2026 19:32:41 by Cesar Gzz
Updated 14 March 2026 19:32:51 by Cesar Gzz