**QueryCloudAZ SDK 1.1**

# Introduction

This document provides a quick guidance on how to use QueryCloudAZ SDK send a wrapped request to NextLabs Authorization REST API and get the response within .net project.

More information about NextLabs Authorization REST API please check Policy Controller help page.

This SDK have controlled the connections to JPC and CC server, so it is safe to use it in multiple threads and the performance is ok.

# Supported Platform

CC version : CC8.6 +

.Net framework: 4.0+

# Important enum and class

## Enum PolicyResult

Description: Query policy results. Include Deny, Allow and DontCare (no policy matched).

Deny: Denied by policy

Allow: Allowed by policy

DontCare: No policy matched

## Enum CEAttributeType

Description: Attribute types.

XacmlString: XACML string

XacmlBoolean: XACML boolean

XacmlInteger: XACML integer

XacmlAnyURI: XACML URI

Others: some types that we don’t care just now.

## Enum QueryStatus

Description: Specify the status of query policy.

S\_OK: Success

E\_Failed: Error, unknown reason

E\_ContentTypeNotSupported: Error, unsupported content type

E\_SendFaild: Error, send request failed, maybe the PC URL is wrong or some other network issue

E\_ResponseStatusAbnormal: Error, get a wrong PC response, abnormal response

E\_TransformResponseFaild: Error, get a wrong PC response, cannot analysis the response

E\_TransformToJsonDataFaild: Error, get a wrong PC response, cannot analysis the response to JSON type

E\_MissAttributes: Error, not all necessary info has been set into the request

E\_Unauthorized: Error, unauthorized from policy control

E\_BadRequest: Error, some unexpected error happened in request

E\_NotFound: Error, the resource cannot fond in cloud AZ

E\_MethodNotAllowed: Error, request method not allow

E\_ProxyAuthenticationRequired: Error, need proxy authentication

E\_RequestTimeout: Error, request time out

E\_RequestUriTooLong: Error, request URL is too long

E\_UnsupportedMediaType: Error, unsupported media type

E\_NotImplemented: Error, function is not implemented

E\_BadGateway: Error, bad gateway

E\_ServiceUnavailable: Error, service unavailable

E\_HttpVersionNotSupported: Error, unsupported HTTP version

E\_GatewayTimeout: Error, gateway timeout

E\_InvalidClient: Error, invalid client

## Class CEAttribute

Description: attribute information.

* CEAttribute(string strName, string strValue, CEAttributeType emAttributeType) Description: Attribute constructor, establish CEAttribute.

strName: Attribute key

strValue: Attribute value

emAttributeType: Attribute type

* Name: Attribute key
* Value: Attribute value
* Type: Attribute type

## Class CEAttres

Description: collection of CEAttribute objects.

* CEAttres ()

Description: Attributes constructor, establish CEAttres.

* void AddAttribute(CEAttribute ceAttr)

Description: Add new CEAttribute to end of collection.

ceAttr: CEAttribute object

* CEAttribute this[int nIndex]

Description: Get CEAttribute by index.

nIndex: Attributes index, start from 0

Return: Return CEAttribute object

* Count : Get CEAttribute objects count

Description: Get CEAttribute objects count.

## Class CERequest

Description: Request object.

* CERequest ()

Description: Constructor, establish CERequest.

* void SetAction(string strAction)

Description: Set action information.

strAction: Action name

* void SetApp(string strName, string strPath, string strUrl, CEAttres ceAttres)

Description: Set application information.

strName: Application name

strPath: Application path

strUrl: Application URL

ceAttres: Application attributes

* void SetUser(string strSid, string strName, CEAttres ceAttres)

Description: Set user information.

strSid: User SID

strName: User name

ceAttres: User attribute

* void SetHost(string strName, string strIPAddress, CEAttres ceAttres)

Description: Set host information.

strName: Host name

strIPAddress: Host IP address

ceAttres: Host attribute

* void SetSource(string strSourceName, string strSourceType, CEAttres ceAttres) Description: Set resource information.

strSourceName: Resource name or resource URL

strSourceType: Resource type (Policy mode short name)

ceAttres: Resource attribute

* void SetDest(string strDestName, string strDestType, CEAttres ceAttres) Description: Set destination information.

strDestName: Destination name or destination URL

strDestType: Destination type (Policy mode short name)

ceAttres: Destination attribute

* void SetEnvAttributes(CEAttres ceAttrs)

Description: Set environment information.

ceAttres: Environment attributes

## Class CEObligation

Description: Obligation object.

* CEAttres GetCEAttres()

Description: Get obligation attributes information.

Return: Obligation attributes information

* string GetName()

Description: Get obligation name.

Return: Obligation name

* string GetPolicyName()

Description: Get matched policy name.

Return: policy name

## Class CEQuery

Description: Main object for query policy.

* CEQuery(string strPCHost, string strOAuthServiceHost, string strClientId, string strClientSecret, int nTokenExpiresTime = 10)

Description: Constructor, establish CEQuery.

strPCHost: Policy control host URL, like http://jpc.crm.nextlabs.solutions:58080

strOAuthServiceHost: OAUTH server host URL, like <https://cc.crm.nextlabs.solutions>

strClientId: Client ID, for example apiuser

strClientSecret: Client secret number, for example 123456

nTokenExpiresTime: Optional parameter, set token expires time, unit is minute

* MultipleLimited : Set limited number of CERequest list for CheckMultipleResources

Description: Set limited number of CERequest list for CheckMultipleResources. (Max is "1000", default is "100")

* Authenticated: Get authenticated status

Description: Get authenticated status.

* void RefreshToken()

Description: Refresh token when the token is expired.

* QueryStatus CheckResource(CERequest ceRequest, out PolicyResult emPolicyResult, out List<CEObligation> listObligation)

Description: Single Query Policy by request information and return policy result and obligations.

ceRequest: Request contains all the information to do query policy.

emPolicyResult : Policy result return from Policy Controller.

listObligation: All obligations return from Policy Controller.

Return: Query policy status

* QueryStatus CheckMultipleResources(List<CERequest> ceListRequests, out List<PolicyResult> listPolicyResults, out List<List<CEObligation>> listObligations)

Description: Multiple query policy by requests information and return policy results and obligations.

ceListRequests: All requests to do query policy.

listPolicyResults: All policy results return from Policy Controller.

listObligations: All obligations return from Policy Controller.

Return: Query policy status

# Thread Safety

* User wants to use CEQuery in thread instead of share CEQuery object to multiple threads. Performance has been optimized already.
* Any instance members are not guaranteed to be thread safe.

# Code example

Start Visual Studio, select file->new->project “Visual C#: Console App”, create project with specified name. Replace file “Program.cs” using below file. Select “References” from project and add “QueryCloudAZSDK.dll” to it. Then build this project to test or debug.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using QueryCloudAZSDK;

using QueryCloudAZSDK.CEModel;

namespace QueryCloudAZSDKTester

{

class Program

{

static CERequest CreateQueryRequest(string strAction, string strUrl)

{

CERequest obCERequest = new CERequest();

// Set action info, this is mandatory

{

obCERequest.Set\_Action(strAction);

}

// Set user info, this is mandatory

{

CEAttres obCEUserAttres = new CEAttres();

obCEUserAttres.Add\_Attre("City", "HangZhou", CEAttributeType.XACML\_String);

obCEUserAttres.Add\_Attre("emailaddress", "george@test.com", CEAttributeType.XACML\_String);

obCERequest.Set\_User("S-1-5-21-310440588-250036847-580389505-500", "Test@nextlabs.com", obCEUserAttres);

}

// Set source info, this is mandatory

{

CEAttres obCESourceAttres = new CEAttres();

obCESourceAttres.Add\_Attre("age", "20", CEAttributeType.XACML\_String);

obCESourceAttres.Add\_Attre("url", strUrl, CEAttributeType.XACML\_String);

// Use "spe" as resoure type (must be same with policy mode name)

obCERequest.Set\_Source(strUrl, "spe", obCESourceAttres);

}

// Set Destination info, this is optional

{

CEAttres obCEDestAttres = new CEAttres();

obCEDestAttres.Add\_Attre("toattr", "dest", CEAttributeType.XACML\_String);

obCERequest.Set\_Dest("C:/Temp/Dest.txt", "spe", obCEDestAttres);

}

// Set application info, this is optional

{

obCERequest.Set\_App("Sharepoint Policy Enforcement", null, null, null);

}

// This method can set NameAttributes (Environmental), this is optional

{

// You must set this if you want get policy result "Dont Care", it means there are no any policy matched.

obCERequest.Set\_EnvAttributes("dont-care-acceptable", "yes", CEAttributeType.XACML\_String);

// You must set this if you want ignore the policy mode name.

obCERequest.Set\_EnvAttributes("use\_resource\_type\_when\_evaluating", "false", CEAttributeType.XACML\_String);

}

// this method can set host, this is optional

{

obCERequest.Set\_Host("HostName", "10.23.60.12", null);

}

return obCERequest;

}

static void Main(string[] args)

{

try

{

// Establish CEQuery object

string strJPCHost = "http://cc87-jpc.xxx.com:58080";

string strOAuthHost = "https://cc87-console.xxx.com";

string strClientId = "apiclient";

string strClientSecure = "xxxxxx";

CEQuery obCEQuery = new CEQuery(strJPCHost, strOAuthHost, strClientId, strClientSecure);

if (QueryStatus.E\_Unauthorized == obCEQuery.Authenticated)

{

// If occur "E\_Unauthorized"，please refresh token again.

obCEQuery.RefreshToken();

}

if (QueryStatus.S\_OK == obCEQuery.Authenticated)

{

List<CERequest> listRequest = new List<CERequest>();

List<PolicyResult> results = null;

PolicyResult resultSingle = PolicyResult.DontCare;

List<QueryCloudAZSDK.CEModel.CEObligation> obSingle = null;

List<List<QueryCloudAZSDK.CEModel.CEObligation>> obligations = null;

// Create CERequest

CERequest requestMy = CreateQueryRequest("OPEN", "Sharepoint://george-sp16/myTestLib/11.docx");

CERequest requestYou = CreateQueryRequest("OPEN", "Sharepoint://george-sp16/youTestLib/22.pptx");

CERequest requestHe = CreateQueryRequest("OPEN", "Sharepoint://daniel-sp16/heTestLib/33.xlsx");

listRequest.Add(requestMy);

listRequest.Add(requestYou);

listRequest.Add(requestHe);

// Do Single Query Policy and get result and obligations

QueryStatus statueSingle = obCEQuery.CheckResource(requestMy, out resultSingle, out obSingle);

if (statueSingle == QueryStatus.S\_OK)

{

// resultSingle PolicyResult

if (resultSingle == PolicyResult.Allow)

{

Console.WriteLine("CheckResource Policy Result: Access is allowed by policy.");

if (obSingle != null && obSingle.Count > 0)

{

foreach (CEObligation ob in obSingle)

{

// obligation information

Console.WriteLine("CheckResource Obligation name: " + ob.Get\_Name());

Console.WriteLine("CheckResource Obligation policy name: " + ob.Get\_PolicyName());

CEAttres obAttrs = ob.GetCEAttres();

int count = obAttrs.get\_count();

for (int i = 0; i < count; i++)

{

string key = "";

string value = "";

CEAttributeType emAttrType = CEAttributeType.XACML\_String;

obAttrs.Get\_Attre(i, out key, out value, out emAttrType);

Console.WriteLine("CheckResource Obligation attribute, key: " + key);

Console.WriteLine("CheckResource Obligation attribute, value: " + value);

Console.WriteLine("CheckResource Obligation attribute, type: " + emAttrType);

}

}

}

}

else if (resultSingle == PolicyResult.Deny)

{

Console.WriteLine("CheckResource Policy Result: Access is denied by policy.");

}

else

{

Console.WriteLine("CheckResource Policy Result: Don't match any policy.");

}

}

QueryStatus statueMultiple = obCEQuery.CheckMultipleResources(listRequest, out results, out obligations);

Console.WriteLine("CheckMultipleResources query policy status: " + statueMultiple.ToString());

if (statueMultiple == QueryStatus.S\_OK)

{

for (int i = 0; i < results.Count; i++)

{

Console.WriteLine("CheckMultipleResources index: " + i + ", policy result: " + results[i].ToString());

if (results[i] == PolicyResult.Allow)

{

if (obligations != null && obligations.Count > i && obligations[i] != null && obligations[i].Count > 0)

{

// obligation information

foreach (CEObligation ob in obligations[i])

{

Console.WriteLine("CheckMultipleResources Obligation name: " + ob.Get\_Name());

Console.WriteLine("CheckMultipleResources Obligation policy name: " + ob.Get\_PolicyName());

CEAttres obAttrs = ob.GetCEAttres();

int count = obAttrs.get\_count();

for (int j = 0; j < count; j++)

{

string key = "";

string value = "";

CEAttributeType emAttrType = CEAttributeType.XACML\_String;

obAttrs.Get\_Attre(j, out key, out value, out emAttrType);

Console.WriteLine("CheckMultipleResources Obligation attribute, key: " + key);

Console.WriteLine("CheckMultipleResources Obligation attribute, value: " + value);

Console.WriteLine("CheckMultipleResources Obligation attribute, type: " + emAttrType);

}

}

}

}

}

}

}

else

{

Console.WriteLine(string.Format("Establish CEQuery object failed: [{0}]\n", obCEQuery.Authenticated));

}

}

catch(Exception exp)

{

Console.WriteLine("Main Exception: " + exp);

}

Console.ReadLine();

}

}

}

## Test case to get query policy result and obligation using above file:

ID 303 STATUS APPROVED POLICY "ROOT\_303/geo-setcolumn"

ATTRIBUTE TRUE\_ALLOW

TAG spe="spe"

FOR TRUE

ON OPEN

BY TRUE

WHERE (TRUE AND (TRUE AND resource.spe.url = "sharepoint://geo\*\*11.docx"))

DO allow

ON allow DO SETCOLUMN\_VALUE("Column Name", "itar", "Column Value", "yes", "All Matched Tags", "No", "policy\_model\_id", "22")

ON deny DO log

ID 304 STATUS APPROVED POLICY "ROOT\_304/geo-deny"

FOR TRUE

ON OPEN

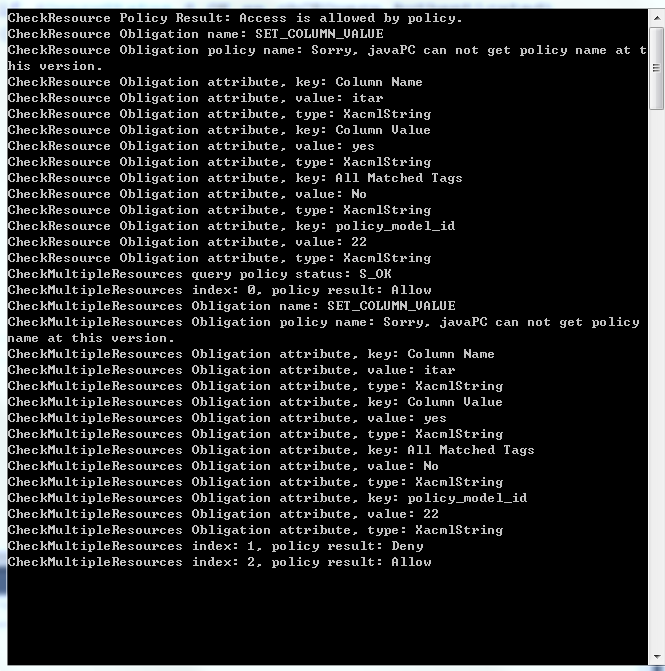
BY TRUE

WHERE (TRUE AND (TRUE AND resource.spe.url = "sharepoint://geo\*\*22.pptx"))

DO deny

BY DEFAULT DO allow

ON deny DO log



# Performance

Cloud AZ Environmental:

|  |  |  |
| --- | --- | --- |
| Region | North Virginia | For Latency parameter |
| Control Center | Instance Type | t2.medium |
|  | RAM | 4.0 GiB |
|  | Number of CPU | 2 vCPU |
|  | Volume Type | EBS (general Purpose SSD) |
|  | Volume Size | 50 GB |
|  | Network | Low to Moderate |
|  | Auto scaling | Architecture doesn’t support scaling – so only 1 instance |
|  |  |  |
| ICE-NET | Instance Type | T2.small |
|  | RAM | 2.0 GiB |
|  | Number of CPU | 1 vCPU |
|  | Volume Type | EBS |
|  | Volume Size | 50 GB |
|  | Network | Low to Moderate |
|  | Scalability | Behind Auto scaling Group |
|  | Scaling Activity | NO – Min and Maximum  Set to 1 instances (meaning it remains 1 regardless of load) |
| JPC | Instance Type | T2.small |
|  | RAM | 2.0 GiB |
|  | Number of CPU | 1 vCPU |
|  | Volume Type | EBS |
|  | Volume Size | 50 GB |
|  | Network | Low to Moderate |
|  | Scalability | Behind Auto scaling Group |
|  | Scaling Activity | NO – Min and Maximum  Set to 2 instances (meaning it remains 2 regardless of load) |
|  |  |  |
| RDS | Database Engine | PostgreSQL 9.5.4 |
|  | IOPS | NOT Supported for this instance type |
|  | Instance Type | db.t2.small |
|  |  | cid:image001.png@01D34E7A.DD404630 |

Windows Instance: Inter(R) Xeon(R) CPU E5-2676 V3 @2.40GHZ 2.40GHZ, 8GB RAM

|  |  |  |
| --- | --- | --- |
| Windows Instance | Instance Type | t2.large |
|  | RAM | 8.0 GiB |
|  | Number of CPU | 2 vCPU |
|  | Volume Type | EBS (general Purpose SSD) |
|  | Volume Size | 40 GB |
|  | Network | Low to Moderate |
|  | OS | Windows server 2012 |

|  |  |
| --- | --- |
| .NET SDK Performance Test Result | |
|  | Response Time |
| Thread |  |
| 1 | 3.884962956 |
| 3 | 4.43567502 |
| 5 | 4.331834059 |
| 7 | 4.033296117 |
| 9 | 4.066425742 |
| 11 | 4.123820555 |
| 13 | 4.192879178 |
| 15 | 3.676241211 |
| 17 | 3.784292021 |
| 19 | 3.945818388 |
| 21 | 4.141557993 |
| 23 | 4.857606752 |
| 25 | 5.108772838 |
| 27 | 5.077417177 |
| 29 | 5.357934096 |
| 31 | 6.439139446 |
| 33 | 6.638609735 |
| 35 | 7.08728129 |
| 37 | 8.184151118 |
| 39 | 8.872921698 |