JCore Client SDK 실행 예제

버전: 1.1.0 | 기준일: 2026-10-16

1. 실행 준비와 순서

이 문서는 접속 설정 → 계정 인증 → 태그 읽기 → 품질과 값 처리 → 종료 순서로 구성했습니다. 01, 02, 03 중 접속 방식 하나를 선택해 Program.cs 전체를 교체하세요. 세 파일을 동시에 복사하면 Program이 중복됩니다. 04는 보조 클래스이고 05는 JCore 서버 스크립트입니다.

Windows Visual Studio에서 콘솔 앱(.NET Framework 4.8, C# 7.3 이상)을 만듭니다. JCoreClient.csproj를 프로젝트 참조하거나 같은 버전 SDK의 JCoreClient.dll, JCore.Contracts.dll 및 종속 DLL을 참조·출력 폴더에 복사합니다. System.Text.Json 참조도 필요합니다. EXE 프로젝트에 아래 속성을 넣어 바인딩 리디렉션을 생성하세요.

```
<PropertyGroup>
  <TargetFrameworkVersion>v4.8</TargetFrameworkVersion>
  <LangVersion>7.3</LangVersion>
  <AutoGenerateBindingRedirects>true</AutoGenerateBindingRedirects>
  <GenerateBindingRedirectsOutputType>true</GenerateBindingRedirectsOutputType>
</PropertyGroup>
```

서버에 Admin 또는 User 계정을 만들고 대상 프로젝트와 태그를 확인합니다. 예제 기본값은 Test.MX46이며 실제 프로젝트·태그명으로 바꾸거나 실행 인수로 지정합니다. 비밀번호는 콘솔에서 입력하므로 코드·명령줄에 저장하지 않습니다. 정상 품질이면 종료 코드 0, 값이 없거나 품질 불량이면 2, 예외면 1입니다. 읽기는 마지막 수집값을 조회하며 PLC를 즉시 새로 스캔하지 않습니다.

2. 로컬 SDK: 인증 후 태그 한 개 읽기

01_SdkLocalRead.cs를 Program.cs로 복사합니다. 실행 인수는 프로젝트명 태그명 파이프명 순서입니다. 예: ClientExample.exe Test MX46 JCore.v1. 서버와 같은 Windows PC에서 실행하며 SDK 라이선스와 Windows 파이프 권한이 필요합니다.

01_SdkLocalRead.cs

```
using System;
using System.Globalization;
using System.Threading;
using System.Threading.Tasks;
using System.Text.Json;
using JCore.Client;
using JCore.Contracts;

internal static class Program
{
    private static int Main(string[] args)
    {
        try { return Run(args).GetAwaiter().GetResult(); }
        catch (Exception e)
        { Console.Error.WriteLine(e.GetType().Name + ": " + e.Message); return 1; }
    }
    private static async Task<int> Run(string[] args)
    {
        string project = args.Length > 0 ? args[0] : "Test";
        string tag = args.Length > 1 ? args[1] : "MX46";
        string pipe = args.Length > 2 ? args[2] : "JCore.v1";
        var core = new JCoreClient(pipe);
        Console.Write("JCore user: "); string user = Console.ReadLine();
        Console.Write("Password: "); string password = ReadPassword();
        bool loggedIn = false;
```

```
        try
        {
            // 1. First request: connect to the local pipe and authenticate.
            await Call(ct => core.LoginAsync(user, password, ct));
            loggedIn = true;
            var identity = await Call(ct => core.GetCurrentUserAsync(ct));
            Console.WriteLine("Authenticated=" + identity.Username + ", Role=" + identity.Role);
            // 2. Name overload uses List + Get. Cache sample.Id for repeated reads.
            TagValue sample = await Call(ct => core.GetTagValueAsync(project, tag, ct));
            // 3. Handle returned quality and value; this does not force a PLC scan.
            bool good = Print(sample.ProjectName + "." + sample.Name, sample.DataType,
                sample.Value, sample.Quality, sample.Timestamp, sample.Error);
            return good ? 0 : 2;
        }
        finally
        {
            password = null;
            if (loggedIn)
            using (var cleanup = new CancellationTokenSource(TimeSpan.FromSeconds(3)))
            try { await core.LogoutAsync(cleanup.Token); }
            catch (Exception e) { Console.Error.WriteLine("Logout: " + e.GetType().Name); }
        }
    }
    private static async Task<T> Call<T>(Func<CancellationToken, Task<T>> action)
    {
        using (var timeout = new CancellationTokenSource(TimeSpan.FromSeconds(15)))
            return await action(timeout.Token);
    }
    private static async Task Call(Func<CancellationToken, Task> action)
    {
        using (var timeout = new CancellationTokenSource(TimeSpan.FromSeconds(15)))
            await action(timeout.Token);
    }
    private static string ReadPassword()
    {
        if (Console.IsInputRedirected) return Console.ReadLine();
        var text = new System.Text.StringBuilder();
        while (true)
        {
            var key = Console.ReadKey(true);
            if (key.Key == ConsoleKey.Enter) { Console.WriteLine(); return text.ToString(); }
            if (key.Key == ConsoleKey.Backspace)
            { if (text.Length > 0) text.Length--; }
            else if (!char.IsControl(key.KeyChar)) text.Append(key.KeyChar);
        }
    }
    private static bool Print(string name, string type, object value,
        string quality, DateTime timestamp, string error)
    {
        Console.WriteLine("Tag=" + name + ", Type=" + type + ", Quality=" + quality);
        Console.WriteLine("Timestamp=" + timestamp.ToString("O"));
        if (quality != "Good" || value == null)
        {
            Console.WriteLine("Value unavailable; do not substitute zero. " + error);
            return false;
        }
        if (value is JsonElement json)
            Console.WriteLine("Value=" + (json.ValueKind == JsonValueKind.String
                ? json.GetString() : json.GetRawText()));
        else Console.WriteLine("Value=" + Convert.ToString(value, CultureInfo.InvariantCulture));
        return true;
    }
}
```

3. 원격 SDK: TLS 검증 후 인증·읽기

02_SdkRemoteRead.cs를 별도 콘솔 앱의 Program.cs로 복사합니다. 실행 인수는 서버IP 포트 인증서SHA256 프로젝트명 태그명입니다. 예: ClientExample.exe 192.168.1.10 4841 서버에서확인한64자리지문 Test MX46. 예의 지문 부분은 실제 값으로 교체해야 합니다. host에는 URL 스킴이나 /api/를 넣지 않습니다. 같은 PC의 파이프 권한 문제를 구분하려면 127.0.0.1과 실제 지문으로 TLS 접속을 테스트할 수 있습니다.

02_SdkRemoteRead.cs

```
using System;
using System.Globalization;
using System.Threading;
using System.Threading.Tasks;
using System.Text.Json;
using JCore.Client;
using JCore.Contracts;

internal static class Program
{
    private static int Main(string[] args)
    {
        try { return Run(args).GetAwaiter().GetResult(); }
        catch (Exception e)
        { Console.Error.WriteLine(e.GetType().Name + ": " + e.Message); return 1; }
    }
    private static async Task<int> Run(string[] args)
    {
        if (args.Length < 5)
            throw new ArgumentException("Usage: SdkRemoteRead host port sha256 project tag");
        string project = args[3], tag = args[4];
        var core = new JCoreClient(args[0], int.Parse(args[1]), args[2]);
        Console.Write("JCore user: "); string user = Console.ReadLine();
        Console.Write("Password: "); string password = ReadPassword();
        bool loggedIn = false;
        try
        {
            // 1. First network call: connect, remote TLS verification, and login.
            await Call(ct => core.LoginAsync(user, password, ct));
            loggedIn = true;
            var identity = await Call(ct => core.GetCurrentUserAsync(ct));
            Console.WriteLine("Authenticated=" + identity.Username + ", Role=" + identity.Role);
            // 2. Name overload uses List + Get. Cache sample.Id for repeated reads.
            TagValue sample = await Call(ct => core.GetTagValueAsync(project, tag, ct));
            // 3. Handle returned quality and value; this does not force a PLC scan.
            bool good = Print(sample.ProjectName + "." + sample.Name, sample.DataType,
                sample.Value, sample.Quality, sample.Timestamp, sample.Error);
            return good ? 0 : 2;
        }
        finally
        {
            password = null;
            if (loggedIn)
            using (var cleanup = new CancellationTokenSource(TimeSpan.FromSeconds(3)))
            try { await core.LogoutAsync(cleanup.Token); }
            catch (Exception e) { Console.Error.WriteLine("Logout: " + e.GetType().Name); }
        }
    }
    private static async Task<T> Call<T>(Func<CancellationToken, Task<T>> action)
    {
        using (var timeout = new CancellationTokenSource(TimeSpan.FromSeconds(15)))
            return await action(timeout.Token);
    }
    private static async Task Call(Func<CancellationToken, Task> action)
    {
        using (var timeout = new CancellationTokenSource(TimeSpan.FromSeconds(15)))
            await action(timeout.Token);
```

```
    }
    private static string ReadPassword()
    {
        if (Console.IsInputRedirected) return Console.ReadLine();
        var text = new System.Text.StringBuilder();
        while (true)
        {
            var key = Console.ReadKey(true);
            if (key.Key == ConsoleKey.Enter) { Console.WriteLine(); return text.ToString(); }
            if (key.Key == ConsoleKey.Backspace)
            { if (text.Length > 0) text.Length--; }
            else if (!char.IsControl(key.KeyChar)) text.Append(key.KeyChar);
        }
    }
    private static bool Print(string name, string type, object value,
        string quality, DateTime timestamp, string error)
    {
        Console.WriteLine("Tag=" + name + ", Type=" + type + ", Quality=" + quality);
        Console.WriteLine("Timestamp=" + timestamp.ToString("O"));
        if (quality != "Good" || value == null)
        {
            Console.WriteLine("Value unavailable; do not substitute zero. " + error);
            return false;
        }
        if (value is JsonElement json)
            Console.WriteLine("Value=" + (json.ValueKind == JsonValueKind.String
                ? json.GetString() : json.GetRawText()));
        else Console.WriteLine("Value=" + Convert.ToString(value, CultureInfo.InvariantCulture));
        return true;
    }
}
```

4. JView: 게시 확인 → 세션 생성 → 배치 읽기 → 종료

먼저 웹에서 대상 태그를 게시합니다. 관리 HTTP 순서는 API 문서 7절을 참고하세요. 이후 03_JViewRead.cs를 별도 콘솔 프로젝트에 넣습니다. 인수는 원격 SDK와 같습니다. JView 라이선스가 필요하며 일반 SDK 라이선스는 이 흐름에 필요하지 않습니다. SDK LoginAsync를 먼저 호출하지 않습니다. OpenJViewSessionAsync가 사용자 인증과 접속자 등록을 수행합니다.

JViewTagPath로 프로젝트/태그 이름을 경로로 만들고, 반환된 Token으로 읽습니다. Token 자체를 출력하지 않습니다. 장시간 화면은 60초 만료 전에 갱신하고 같은 세션을 재사용합니다. 아래는 한 번 읽고 즉시 닫는 예제입니다.

03_JViewRead.cs

```
using System;
using System.Globalization;
using System.Threading;
using System.Threading.Tasks;
using System.Text.Json;
using JCore.Client;
using JCore.Contracts;

internal static class Program
{
    private static int Main(string[] args)
    {
        try { return Run(args).GetAwaiter().GetResult(); }
        catch (Exception e)
        { Console.Error.WriteLine(e.GetType().Name + ": " + e.Message); return 1; }
    }
    private static async Task<int> Run(string[] args)
    {
        if (args.Length < 5)
            throw new ArgumentException("Usage: JViewRead host port sha256 project tag");
        var core = new JCoreClient(args[0], int.Parse(args[1]), args[2]);
        string path = JCoreClient.JViewTagPath(args[3], args[4]);
```

```
        Console.Write("JCore user: "); string user = Console.ReadLine();
        Console.Write("Password: "); string password = ReadPassword();
        JViewSession lease = null;
        try
        {
            // 1. Connect + verify TLS pin + authenticate + reserve one JView seat.
            // Do not call SDK LoginAsync first. Publish the tag before this call.
            lease = await Call(ct => core.OpenJViewSessionAsync(user, password, ct));
            Console.WriteLine("Authenticated=" + lease.User.Username);
            Console.WriteLine("Limits=" + lease.MaxTags + " tags / "
                + lease.MaxRuntimeSessions + " seats");
            // 2. Read the published tag by name. Token is never printed.
            var values = await Call(ct => core.ReadJViewTagsAsync(
                lease.Token, new[] { path }, ct));
            // 3. Check and handle the actual returned sample.
            var sample = values[0];
            return Print(sample.Path, sample.DataType, sample.Value,
                sample.Quality, sample.Timestamp, sample.Error) ? 0 : 2;
        }
        finally
        {
            password = null;
            if (lease != null)
            using (var cleanup = new CancellationTokenSource(TimeSpan.FromSeconds(3)))
            try { await core.CloseJViewSessionAsync(lease.Token, cleanup.Token); }
            catch (Exception e) { Console.Error.WriteLine("Close: " + e.GetType().Name); }
        }
    }
    private static async Task<T> Call<T>(Func<CancellationToken, Task<T>> action)
    {
        using (var timeout = new CancellationTokenSource(TimeSpan.FromSeconds(15)))
            return await action(timeout.Token);
    }
    private static async Task Call(Func<CancellationToken, Task> action)
    {
        using (var timeout = new CancellationTokenSource(TimeSpan.FromSeconds(15)))
            await action(timeout.Token);
    }
    private static string ReadPassword()
    {
        if (Console.IsInputRedirected) return Console.ReadLine();
        var text = new System.Text.StringBuilder();
        while (true)
        {
            var key = Console.ReadKey(true);
            if (key.Key == ConsoleKey.Enter) { Console.WriteLine(); return text.ToString(); }
            if (key.Key == ConsoleKey.Backspace)
            { if (text.Length > 0) text.Length--; }
            else if (!char.IsControl(key.KeyChar)) text.Append(key.KeyChar);
        }
    }
    private static bool Print(string name, string type, object value,
        string quality, DateTime timestamp, string error)
    {
        Console.WriteLine("Tag=" + name + ", Type=" + type + ", Quality=" + quality);
        Console.WriteLine("Timestamp=" + timestamp.ToString("O"));
        if (quality != "Good" || value == null)
        {
            Console.WriteLine("Value unavailable; do not substitute zero. " + error);
            return false;
        }
        if (value is JsonElement json)
            Console.WriteLine("Value=" + (json.ValueKind == JsonValueKind.String
                ? json.GetString() : json.GetRawText()));
        else Console.WriteLine("Value=" + Convert.ToString(value, CultureInfo.InvariantCulture));
        return true;
    }
}
```

5. 로그인 이후 재사용하는 보조 함수

04_ClientOperations.cs는 별도 파일로 프로젝트에 추가합니다. 이 파일 자체에는 Main이 없습니다. 01 또는 02의 로그인 성공 후, 로그아웃하기 전에 아래 함수들을 필요한 것만 호출합니다. INT32 Memory 태그의 ID를 읽어 ReadInt32Async와 WriteOnceAsync에 전달하세요. 쓰기 예제는 확인된 Memory 테스트 태그에서만 먼저 사용합니다.

호출 순서: await core.LoginAsync(user, password, ct) → var sample = await core.GetTagValueAsync(project, memoryTag, ct) → await ClientOperations.ReadInt32Async(core, sample.Id, ct). 쓰기는 await ClientOperations.WriteOnceAsync(core, sample.Id, 123, ct). 각 반환 상태를 확인하고 최종적으로 LogoutAsync를 호출합니다. 함수별 정확한 반환 타입과 인수는 코드에 있습니다.

WatchForTenSecondsAsync는 자체 10초 취소를 처리합니다. 이를 15초짜리 단일 요청 래퍼와 혼동하지 마세요. ReadJViewForThirtySecondsAsync는 03의 lease를 만든 뒤 finally 전에 호출하고, 화면 수명용 취소 토큰을 전달합니다. 이 함수는 20초마다 임대를 갱신하며 종료 시 lease 소유자가 CloseJViewSessionAsync를 호출해야 합니다. 갱신 토큰은 현재 서버에서 같은 세션 토큰을 유지합니다.

알람 이력 예제는 GetDataProjectsAsync → ProjectId 선택 → UTC 범위·Limit 지정 → 페이지 조회 → StorageStatus/HasMore 검사 순서입니다. 로깅 이력은 같은 DataQuery로 GetTagHistoryAsync를 호출합니다. 알람 변경은 Epoch/Sequence와 ResyncRequired를 처리하거나 WatchAlarmsAsync를 사용합니다.

04_ClientOperations.cs

```
using System;
using System.Linq;
using System.Threading;
using System.Threading.Tasks;
using JCore.Client;
using JCore.Contracts;

public static class ClientOperations
{
    // Call after SDK LoginAsync; id is the Id of an INT32 sample.
    public static async Task<int> ReadInt32Async(
        JCoreClient core, Guid id, CancellationToken ct)
    {
        var sample = await core.GetTagValueAsync(id, ct);
        if (sample.Quality != "Good" || sample.Value == null)
            throw new InvalidOperationException(sample.Quality + ": " + sample.Error);
        return JCoreClient.GetValue<int>(sample);
    }

    // Use a writable INT32 Memory tag for a safe first test.
    public static async Task<WriteResult> WriteOnceAsync(
        JCoreClient core, Guid id, int value, CancellationToken ct)
    {
        var result = await core.SetTagValueAsync(id, value, ct);
        Console.WriteLine("Write status: " + result.Status);
        // Unknown means the operation may already have reached the server.
        // Do not automatically retry PLC writes.
        return result;
    }

    public static async Task WatchForTenSecondsAsync(JCoreClient core, Guid id)
    {
        using (var stop = new CancellationTokenSource(TimeSpan.FromSeconds(10)))
        try
        {
            await core.SubscribeAsync(new[] { id }, samples =>
            {
                foreach (var sample in samples)
                    Console.WriteLine(sample.Name + ": " + sample.Quality);
                return Task.CompletedTask;
            }, stop.Token);
```

```
        }
        // .NET Framework can surface a closed transport exception during cancellation.
        // Suppress it only after our own lifetime token has been cancelled.
        catch (Exception) when (stop.IsCancellationRequested) { }
    }

    public static async Task AlarmHistoryAsync(
        JCoreClient core, string projectName, CancellationToken ct)
    {
        var projects = await core.GetDataProjectsAsync(ct);
        var project = projects.Single(p => string.Equals(
            p.Name, projectName, StringComparison.OrdinalIgnoreCase));
        var query = new DataQuery
        {
            ProjectId = project.Id,
            FromUtc = DateTime.UtcNow.AddDays(-1), ToUtc = DateTime.UtcNow,
            Limit = 100
        };
        while (true)
        {
            var page = await core.GetAlarmHistoryAsync(query, ct);
            if (page.StorageStatus != "Good")
                throw new InvalidOperationException(page.StorageStatus);
            foreach (var row in page.Items)
                Console.WriteLine(row.Timestamp.ToString("O") + " "
                    + row.Group + " " + row.TagName + " " + row.Kind);
            if (!page.HasMore) break;
            if (page.NextOffset <= query.Offset)
                throw new InvalidOperationException("Invalid pagination progress.");
            query.Offset = page.NextOffset;
        }
    }

    public static async Task<object> CallAddAsync(
        JCoreClient core, CancellationToken ct)
    {
        var result = await core.InvokeScriptAsync(
            "Test", "Calculator", "Add", new object[] { 2, 3 }, ct);
        if (result.Status != "Completed")
            throw new InvalidOperationException(result.Status + ": " + result.Error);
        return result.Value;
    }

    // Caller owns the lease and must close it in finally.
    // Use a screen-lifetime token, not a single 15-second request timeout.
    public static async Task ReadJViewForThirtySecondsAsync(
        JCoreClient core, JViewSession lease, string path, CancellationToken ct)
    {
        var end = DateTime.UtcNow.AddSeconds(30);
        var renewAt = DateTime.UtcNow.AddSeconds(20);
        while (DateTime.UtcNow < end)
        {
            ct.ThrowIfCancellationRequested();
            if (DateTime.UtcNow >= renewAt)
            {
                lease = await core.RenewJViewSessionAsync(lease.Token, ct);
                renewAt = DateTime.UtcNow.AddSeconds(20);
            }
            var samples = await core.ReadJViewTagsAsync(
                lease.Token, new[] { path }, ct);
            Console.WriteLine(samples[0].Path + ": " + samples[0].Quality);
            await Task.Delay(1000, ct);
        }
    }
}
```

6. 외부 메서드 호출용 서버 스크립트

이 파일은 콘솔 앱이 아니라 JCore 웹의 Test 프로젝트 → Scripts → Calculator에 등록합니다. 활성화 후 저장하고 정상 상태를 확인하세요. 외부 클라이언트는 04의 CallAddAsync를 로그인 후 호출합니다. 결과 Status가 Completed이면 Value가 5입니다. 반환 Value는 JSON 값일 수 있으므로 DTO 타입을 확인해 변환합니다. 서버 등록 스크립트명과 InvokeScriptAsync의 두 번째 인수가 반드시 같아야 합니다.

05_CalculatorScript.cs

```
using System.Threading;
using System.Threading.Tasks;
using JCore.Scripting;

// Server-side script: create Test > Scripts > Calculator, then paste this code.
public class Calculator : JCoreScript
{
    public override void Initialize() { Log.Info("Calculator initialized"); }
    public override Task ExecuteAsync(CancellationToken ct)
    {
        return Task.CompletedTask;
    }

    [ScriptCallable(AllowExternal = true)]
    public int Add(int a, int b) { return a + b; }
}
```

7. 운영 전 확인

전체 공개 함수의 반환 타입과 인수는 API 문서 9절에 있습니다. 예제의 원격 주소·계정·태그·인증서 지문은 자신의 환경에 맞춥니다. 예제에서 로그인·읽기는 각각 독립적인 15초 취소 범위를 사용합니다. 서버 인증서 변경 후에는 새 지문을 신뢰할 수 있는 경로로 확인하고 갱신하세요.

Quality=Good인지 확인하기 전 값을 사용하지 않습니다. 통신 실패를 값 0이나 서버가 보낸 ConnectionLost로 꾸미지 않습니다. WriteResult.Unknown이나 ScriptResult.Unknown은 이미 실행됐을 수 있으므로 자동 재시도하지 않습니다. 실제 PLC 쓰기 전에 Memory 태그에서 확인하세요. 일반 SDK SubscribeAsync는 전체 목록 폴링이므로 대규모 화면에서 선택 태그 배치 읽기와 혼동하지 않습니다.

JView 이력 조회는 같은 lease.Token과 new JViewDataQuery { ProjectName = project, TagPath = path, From = DateTimeOffset.UtcNow.AddDays(-1), To = DateTimeOffset.UtcNow, Limit = 100 }을 GetJViewTagHistoryAsync 또는 GetJViewAlarmHistoryAsync에 전달합니다. StorageStatus와 HasMore/NextOffset을 확인하고 이벤트 스트림은 Epoch/Sequence를 유지합니다. 이력은 설정된 보관 기간 안에서만 조회됩니다.