JCore Client SDK API Reference

버전: 1.1.0 | 기준일: 2026-10-16 | 대상: Windows / .NET Framework 4.8

1. 시작하기

JCore.Client.JCoreClient는 일반 SDK와 JView 런타임의 통신 클라이언트입니다. 일반 SDK, OPC UA, JView 라이선스는 각각 독립적으로 검사합니다. 이 문서는 실제 1.1.0 공개 메서드를 기준으로 작성했습니다.

Visual Studio에서 콘솔 앱(.NET Framework)을 만들고 .NET Framework 4.8을 선택합니다. C# 7.3 이상을 사용합니다. 소스가 있으면 JCoreClient/JCoreClient.csproj를 프로젝트 참조로 추가합니다. 배포 SDK를 사용하면 같은 빌드의 JCoreClient.dll, JCore.Contracts.dll 및 함께 제공된 종속 DLL을 참조·복사하세요. System.Text.Json도 필요합니다. 서버 설치 폴더만으로 개발용 SDK가 구성되는 것은 아닙니다.

실행 EXE 프로젝트에서 AutoGenerateBindingRedirects와 GenerateBindingRedirectsOutputType을 true로 설정합니다. 여러 예제의 Program 클래스를 한 프로젝트에 동시에 넣지 마세요. JCoreClient 자체에는 IDisposable, Connect, Disconnect, IsConnected가 없습니다. 생성자는 설정만 보관하며 각 요청이 연결을 열고 응답 후 닫습니다.

접속 종류	주소 예	인증 순서
로컬 일반 SDK	JCore.v1 (Windows Named Pipe)	LoginAsync → GetCurrentUserAsync → 읽기/쓰기 → LogoutAsync
원격 일반 SDK	서버 IP + 4841 + 인증서 SHA256	첫 요청에서 TCP/TLS 검증 → LoginAsync → 읽기/쓰기 → LogoutAsync
JView 런타임	서버 IP + 4841 + 인증서 SHA256	OpenJViewSessionAsync → 읽기/쓰기·갱신 → CloseJViewSessionAsync
웹 관리 API	http://localhost:8080/api/	auth/login → X-JCore-Token 헤더 → 관리 요청 → auth/logout
OPC UA	opc.tcp://서버:4840/JOpcCore	OPC UA 클라이언트의 인증·인증서 절차 사용

원격 SDK 생성자의 host에는 IP 또는 호스트명만 넣습니다. http://, opc.tcp://, /api/는 넣지 않습니다. SHA256은 서버 인증서의 64자리 16진수 지문이며 로그인 토큰이 아닙니다. 콜론·하이픈·공백은 제거하여 비교합니다. TLS 1.2로 인증서 지문과 유효기간을 검사합니다. 서버 인증서가 바뀌면 신뢰할 수 있는 경로로 새 지문을 확인해 갱신하세요.

2. 인증·권한·라이선스

웹과 서비스는 같은 사용자 저장소를 사용하며 사용자 ID 비교는 대소문자를 구분하지 않습니다. Admin/User 계정을 사용하고 비밀번호는 실행 시 입력하거나 애플리케이션 비밀 저장소에서 가져오세요. 비밀번호·토큰을 로그에 남기지 마세요.

일반 SDK는 인증 모드에서도 로그인 전 읽기가 허용되는 기존 정책을 유지합니다. SDK 라이선스와 Windows 파이프 접근 권한 검사는 그대로 적용됩니다. 인증 모드 ON이면 쓰기에 계정 인증이 필요하고 OFF이면 쓰기 가능한 태그에 미인증 쓰기가 허용됩니다. 예제는 환경에 관계없이 명시적으로 로그인합니다. GetCurrentUserAsync는 로그인 후 사용합니다.

SDK 로그인 토큰은 별도의 시간 만료가 없지만 로그아웃, 서버 재시작, 계정 삭제·비활성화·암호/권한 변경 시 재인증이 필요합니다. 웹 Guest 토큰은 30분입니다. 웹 토큰, 일반 SDK 토큰, JView 세션 토큰은 서로 교환할 수 없습니다. 같은 클라이언트 인스턴스에서 로그인/로그아웃을 동시에 실행하지 마세요.

JView는 인증 모드 OFF에서도 Admin/User 인증이 필요합니다. Guest는 제외됩니다. JView 세션은 60초 임대이며 만료 전에 RenewJViewSessionAsync를 호출합니다(예: 20초마다). 연결당 매번 새 세션을 생성하지 말고 런타임 단위로 재사용한

뒤 종료 시 닫습니다. 서버 재시작 후 새 세션이 필요합니다. 여러 브라우저의 접속자 제한을 우회하도록 하나의 세션을 공유하지 마세요.

OPC/SDK에서 MaxTags=null은 무제한입니다. JView의 MaxTags와 MaxRuntimeSessions는 양의 정수이며 null 무제한 방식이 아닙니다. 설치 평가판은 7일, OPC 무제한 / SDK 1,000태그 / JView 1,000태그·5접속입니다. 라이선스 파일 등록·삭제 후에는 전체 서버 재시작으로 반영합니다. OPC UA 설정 재적용만으로는 라이선스를 다시 읽지 않습니다.

3. 일반 SDK 태그 API

메서드	반환 및 사용법
LoginAsync / LogoutAsync	Task. 계정 인증 / 토큰 해제
GetCurrentUserAsync	AuthenticatedUser: Id, Username, Role
GetTagsAsync	TagValue[]. 모든 프로젝트의 현재 태그 목록
GetTagValueAsync(Guid)	TagValue. 캐시된 최신 수집값 조회
GetTagValueAsync(projectName, tagName)	TagValue. 전체 목록 조회 후 이름 검색·ID 읽기
SetTagValueAsync(Guid, object)	WriteResult. 가능한 태그에 한 번 쓰기
SetTagValueAsync(projectName, tagName, object)	이름 확인 후 쓰기. 반복 작업에는 ID 캐시 권장
GetValue<T>(TagValue)	T. JSON 또는 CLR 값을 지정 타입으로 변환; 품질은 직접 확인
SubscribeAsync	Task. 선택 ID의 변경값을 비동기 콜백으로 전달

태그 이름은 프로젝트 안에서 대소문자 무시로 고유합니다. 서로 다른 프로젝트에는 같은 이름을 쓸 수 있으므로 projectName과 tagName을 모두 지정합니다. 일반 SDK의 Id는 내부 GUID이며 CSV 교체로 ID가 달라지면 캐시를 다시 구성합니다. JView에는 아래의 Project.TagName 경로 사용을 권장합니다.

GetTagsAsync에는 페이지 기능이 없고 프레임 최대 크기는 8 MiB입니다. 일반 SDK에는 선택 ID 배열을 한 번에 읽는 공개 배치 메서드가 없습니다. 10만 태그를 매 주기 전체 목록으로 받지 마세요. 목록을 한 번 확인하고 필요한 ID를 캐시하거나 JView의 경로 배치 읽기를 사용합니다. 서버 내부 배치 인덱스가 추가되었다고 일반 SDK의 전체 목록 API가 자동으로 페이지 API가 되는 것은 아닙니다.

SubscribeAsync는 최대 10,000개 ID를 지정하지만 250ms 간격으로 전체 목록을 조회해 변경을 추리는 폴링 방식입니다. 서버 푸시나 모든 변화의 무손실 이벤트 기록을 보장하지 않습니다. 콜백은 UI 스레드가 아니므로 UI 갱신은 별도로 전달합니다. 종료하려면 CancellationToken을 취소하고 정상 취소 예외를 종료로 처리합니다. 알람 이력과 변경 이벤트는 전용 API를 사용하세요.

4. 결과 값·품질·쓰기

TagValue 필드	의미
Id / ProjectName / Name / Path	식별 정보. JViewTagValue에는 Id가 없음
TagClass	Device, Memory, DB, System
DataType / StorageDataType	논리 타입 / 저장 타입. StorageDataType은 일반 SDK DTO에 있음
Value / Quality / Error	값 / 품질 / 오류 원인
Timestamp / StatusTimestamp / Sequence	샘플 시각 / 상태 시각 / 순서
CanWrite	해당 태그 쓰기 가능 여부

읽기는 PLC에 즉시 새 요청을 강제하는 기능이 아니라 코어가 마지막으로 수집한 상태를 반환합니다. Quality가 Good이고 Value가 null이 아닐 때 사용하세요. 통신 예외를 임의로 Good이나 숫자 0으로 바꾸지 마세요. Value는 System.Text.Json.JsonElement로 역직렬화될 수 있습니다. INT32라면 GetValue<int>, INT16은 GetValue<short>, FLOAT는 GetValue<float>, BOOL은 GetValue<bool>, STRING은 GetValue<string>처럼 실제 타입에 맞춥니다. 잘못된 타입 변환은 예외가 됩니다.

System과 DB 태그는 읽기 전용입니다. Memory 태그는 타입에 맞는 값을 쓸 수 있고 PLC는 드라이버와 주소 영역의 쓰기 가능 여부를 따릅니다. DB 값은 첫 행의 컬럼을 CSV 규칙으로 묶은 문자열입니다. Memory 마지막 값 보존은 태그 속성에서 선택하며 기본은 꺼짐입니다.

WriteResult.Status	처리
Acknowledged	프로토콜 응답 확인. 설비 동작 완료 의미는 아님
Rejected / NotSent	거부 또는 전송되지 않음. 원인·권한·타입 확인
QueueFull / QueueTimeout	큐 상태 확인 후 사용자가 재시도 여부 판단
Unknown	이미 적용되었을 수 있음. 자동 재시도 금지, 현재값/설비 상태 확인

WriteResult에는 Status와 nullable bool ReadbackMatches가 있습니다. 드라이버별 상태도 반환될 수 있습니다. 쓰기와 스크립트 호출은 전송 후 통신 취소 시 예외 대신 Unknown 결과가 올 수 있으므로 반환 상태도 검사해야 합니다.

Timestamp는 O 형식으로 출력하면 시간대 정보 확인에 유리합니다. UTC DateTime을 로컬 시각으로 표시할 때 클라이언트 PC 시간대와 서버 시간대가 같다고 가정하지 마세요. JView 이력의 DateTimeOffset은 서버 오프셋을 포함합니다.

5. 알람·로깅·스크립트

메서드	역할
GetDataProjectsAsync	DataProject[]: Id, Name
GetCurrentAlarmsAsync / GetCurrentAlarmCountAsync	현재 알람 목록 / 개수
GetAlarmHistoryAsync	보관된 알람 이력
GetAlarmChangesAsync	Epoch/AfterSequence 이후 알람 변화
WatchAlarmsAsync	최초 현재 상태 및 이후 알람 변경 폴링
GetTagHistoryAsync	로깅을 켜서 저장한 태그 이력
InvokeScriptAsync	외부 호출을 허용한 프로젝트 스크립트 메서드 실행

DataQuery에는 ProjectId, TagId(선택), TagName, FromUtc, ToUtc, Group, Level, Offset, Limit, AfterSequence, Epoch가 있습니다. ProjectId는 GetDataProjectsAsync에서 가져옵니다. UTC 범위를 사용하며 FromUtc ≤ ToUtc, 최대 3,660일, Limit 1~100, Offset 0~100,000입니다. HasMore가 true이면 NextOffset으로 다음 페이지를 조회합니다. StorageStatus가 Good인지 확인해야 빈 목록과 저장소 장애를 구분할 수 있습니다.

DataResult는 Items, Count, HasMore, NextOffset, Epoch, Sequence, ResyncRequired, StorageStatus 및 저장소 용량 정보를 제공합니다. DataRecord에는 태그·프로젝트 ID/이름, Kind, Group, Level, Description, Condition, Timestamp, EndedUtc, Reason, DataType, Value, CurrentValue, Quality, Sequence가 있습니다. 그룹 정보도 포함됩니다. 알람 해제 이유와 품질을 함께 확인하세요.

변경 수신은 Epoch와 Sequence를 보관하며 ResyncRequired이면 현재 알람을 다시 조회합니다. WatchAlarmsAsync는 이 흐름을 처리하고 변경이 없으면 약 500ms 간격으로 확인합니다. 콜백은 Action<DataResult>이므로 async void로 작성하지 마세요. 태그 로깅 이력은 로깅 사용 여부·Change/Interval 설정·보관 기간에 따라 달라집니다.

스크립트 호출은 프로젝트명, 등록된 스크립트명, 메서드명, object[] 인수 순서입니다. 외부 호출 메서드는 public이며 [ScriptCallable(AllowExternal = true)]가 필요합니다. ScriptResult의 Status가 Completed일 때 Value를 사용하고 Failed, Rejected, Disabled, Unknown 등은 별도로 처리합니다. 실패한 PLC 쓰기나 스크립트를 자동 반복 실행하지 마세요.

6. JView 전용 API

메서드	반환 및 역할
JViewTagPath(projectName, tagName)	string. 이름을 안전하게 인코딩한 경로
OpenJViewSessionAsync	JViewSession: Token, User, ExpiresAtUtc, MaxTags, MaxRuntimeSessions
RenewJViewSessionAsync / CloseJViewSessionAsync	세션 갱신 / 접속자 슬롯 반환
ReadJViewTagsAsync(token, string[] paths)	JViewTagValue[]. 게시된 경로 배치 읽기
ReadJViewTagAsync(token, string path)	JViewTagValue. 단일 읽기
WriteJViewTagAsync(token, string path, object value)	WriteResult. 게시된 쓰기 가능 태그
GetJViewCurrentAlarmsAsync	현재 알람
GetJViewAlarmHistoryAsync	알람 이력
GetJViewAlarmEventsAsync	발생·해제 변화 이벤트
GetJViewTagHistoryAsync	태그 저장 이력

JView는 로컬 파이프가 아닌 TLS SDK 포트로 접속합니다. 같은 PC도 127.0.0.1과 인증서 SHA256을 사용합니다. 일반 SDK LoginAsync를 선행하지 않습니다. OpenJViewSessionAsync가 계정 인증과 JView 접속자 등록을 함께 수행합니다.

JViewTagPath("Line.1", "Sensor.Value")는 Line%2E1.Sensor%2EValue를 반환합니다. 단순 문자열 연결 대신 이 함수를 사용하세요. 일반 경로는 Test.MX46처럼 보입니다. Guid[] 읽기와 Guid 쓰기는 호환용이며 신규 JView UI에 UUID를 노출하지 마세요.

JViewDataQuery에는 ProjectName, TagPath, From/To(DateTimeOffset), Group, Level, Offset, Limit, Epoch, AfterSequence가 있습니다. 시간·페이지 제한은 일반 조회와 같습니다. JViewDataResult의 Items는 JViewDataRecord 배열이며 EventKey, TagPath, RecordedTagName, EventType(Raised/Cleared/Sample), Kind, Group, Level, Description, Condition, Timestamp, EndedAt, Reason, DataType, Value, CurrentValue, Quality, Sequence를 제공합니다. 현재 게시 태그에 대한 접근 범위가 적용됩니다.

화면에 필요한 태그를 묶어 읽고 큰 목록은 예를 들어 1,000개씩 나눠 프레임 크기와 실제 값 크기를 고려합니다. 라이선스 수량은 매 요청의 배열 길이만이 아니라 게시된 유효 태그의 중복 제외 총량입니다. 세션은 재사용하고 로그에 Token을 출력하지 마세요.

7. JView 태그 게시용 웹 관리 인터페이스

이 절은 JCoreClient 메서드가 아닌 HTTP 관리 API입니다. JView 서버에서 계정을 인증한 후 게시합니다. 원격 환경은 보호된 HTTPS 관리 주소를 사용하세요. 브라우저에 관리 비밀번호를 내장하지 마세요.

1. POST /api/auth/login, JSON: {"Username":"사용자","Password":"비밀번호"}. 2. 응답 Token을 X-JCore-Token 헤더에 설정합니다. 3. POST /api/jview/tags, JSON: {"TagPaths":["Test.MX46"],"Mode":"Merge"}. 4. GET /api/jview/tags로 TagPaths, TagCount, MissingTagCount, ActiveSessions를 확인합니다. 5. POST /api/auth/logout 후 웹 토큰을 폐기합니다. 런타임에서는 별도 OpenJViewSessionAsync를 사용합니다.

Admin/User는 게시 등록·조회·변경 권한이 같고 Guest는 사용할 수 없습니다. Merge는 기존 게시 목록과 합집합(기본), Remove는 선택 태그 제외, Replace는 전달 목록으로 전체 교체입니다. 기존 게시 태그를 보존하려면 Merge를 사용하세요. TagPaths와 호환용 TagIds를 동시에 보내지 않습니다. 본문은 1 MiB 이내로, 대량 등록은 약 1,000개씩 분할합니다.

삭제·CSV 교체로 존재하지 않는 태그는 게시 수량에서 자동 제외됩니다. 같은 이름이라도 CSV 교체로 내부 ID가 바뀌었다면 다시 게시하세요. 게시 제거는 기존 런타임 세션을 초기화할 수 있으므로 다시 연결합니다. GET /api/license/status에서 Features 중 JVIEW의 License.MaxTags/MaxRuntimeSessions, JViewTagCount와 RestartRequired를 확인할 수 있습니다.

8. 제한시간·장애 처리

SDK 개별 요청의 내부 제한은 15초이고 로컬 파이프 연결 제한은 3초입니다. 호출자의 CancellationToken은 더 짧게 취소할 수 있지만 내부 제한을 늘리지는 않습니다. 로그인과 읽기에 하나의 합산 5초 제한을 공유하지 마세요. 화면 전체 작업 취소와 개별 요청 취소를 구분하세요. .NET Framework에서는 취소 중 연결이 닫히며 전송 예외가 발생할 수도 있으므로, 직접 소유한 종료 토큰이 취소된 경우에만 정상 종료로 처리합니다. 일반 통신 장애는 숨기지 않습니다.

증상	확인 및 처리
UnauthorizedAccessException	계정·쓰기 권한·라이선스·파이프 ACL 확인
AuthenticationException	TLS 지문·인증서 유효기간·서버 인증서 변경 확인
IOException / TimeoutException	서비스 실행, 포트, 방화벽, 디버거 일시정지 확인
OperationCanceledException	사용자 취소인지 요청 시간 초과인지 구분
JVIEW_LICENSE_REQUIRED	JView 기능 라이선스 및 서버 재시작 여부
JVIEW_TAG_LIMIT_EXCEEDED	유효 게시 태그의 병합 후 중복 제외 총량 확인
JVIEW_SESSION_LIMIT_EXCEEDED	닫히지 않은 세션·접속자 수 확인
JVIEW_SESSION_EXPIRED	새 계정 인증으로 세션 재생성
JVIEW_TAG_NOT_PUBLISHED / TAG_NOT_FOUND	게시 목록·프로젝트·태그 존재 여부 확인
JVIEW_TAG_AMBIGUOUS / INVALID_TAG_PATH	경로 생성 함수와 이름 중복 확인
JVIEW_PROJECT_NOT_FOUND / HISTORY_UNAVAILABLE	프로젝트 또는 이력 저장소 상태 확인

위 표의 축약 TAG_NOT_FOUND 등도 실제로는 JVIEW_ 접두사를 포함합니다. 읽기 실패에 대한 재연결은 제한된 정책으로 구성할 수 있지만 쓰기·외부 메서드 호출의 자동 재전송은 금지합니다. 서버 진단 로그의 연결/요청 ID·수신/처리/전송 시각으로 지연 구간을 비교하세요.

9. 전체 공개 함수 서명

아래는 JCoreClient.cs에서 추출한 1.1.0 공개 서명입니다. 반환 타입과 매개변수 순서를 그대로 확인할 수 있습니다. Task는 await해서 사용합니다.

```
public JCoreClient(string host, int port, string serverCertificateSha256);
```

```
public JCoreClient(string pipeName=PipeProtocol.DefaultPipeName);
```

```
public async Task<DataProject[]> GetDataProjectsAsync(CancellationToken ct=default(CancellationToken));
```

```
public async Task<DataResult> GetCurrentAlarmsAsync(DataQuery query,CancellationToken ct=default(CancellationToken));
```

```
public async Task<int> GetCurrentAlarmCountAsync(Guid projectId,CancellationToken ct=default(CancellationToken));
```

```
public Task<DataResult> GetAlarmHistoryAsync(DataQuery query,CancellationToken ct=default(CancellationToken));
```

```
public Task<DataResult> GetTagHistoryAsync(DataQuery query,CancellationToken ct=default(CancellationToken));
```

```
public Task<DataResult> GetAlarmChangesAsync(DataQuery query,CancellationToken ct=default(CancellationToken));
```

```
public async Task WatchAlarmsAsync(Guid projectId,Action<DataResult> changed,CancellationToken ct);
```

```
public async Task<JCore.Scripting.ScriptResult> InvokeScriptAsync(string project,string script,string method,object[]
    arguments,CancellationToken cancellation=default(CancellationToken));
```

```
public static string JViewTagPath(string projectName,string tagName);
```

```
public async Task<JViewTagValue[]> ReadJViewTagsAsync(string token,string[] paths,CancellationToken
    ct=default(CancellationToken));
```

```
public async Task<JViewTagValue> ReadJViewTagAsync(string token,string path,CancellationToken
    ct=default(CancellationToken));
```

```
public async Task<WriteResult> WriteJViewTagAsync(string token,string path,object value,CancellationToken
    ct=default(CancellationToken));
```

```
public Task<JViewDataResult> GetJViewCurrentAlarmsAsync(string token,JViewDataQuery query,CancellationToken
    ct=default(CancellationToken));
```

```
public Task<JViewDataResult> GetJViewAlarmHistoryAsync(string token,JViewDataQuery query,CancellationToken
    ct=default(CancellationToken));
```

```
public Task<JViewDataResult> GetJViewAlarmEventsAsync(string token,JViewDataQuery query,CancellationToken
    ct=default(CancellationToken));
```

```
public Task<JViewDataResult> GetJViewTagHistoryAsync(string token,JViewDataQuery query,CancellationToken
    ct=default(CancellationToken));
```

```
public async Task<JViewSession> OpenJViewSessionAsync(string username,string password,CancellationToken
    ct=default(CancellationToken));
```

```
public async Task<JViewSession> RenewJViewSessionAsync(string token,CancellationToken ct=default(CancellationToken));
```

```
public async Task CloseJViewSessionAsync(string token,CancellationToken ct=default(CancellationToken));
```

```
public async Task<TagValue[]> ReadJViewTagsAsync(string token,Guid[] ids,CancellationToken
    ct=default(CancellationToken));
```

```
public async Task<WriteResult> WriteJViewTagAsync(string token,Guid id,object value,CancellationToken
    ct=default(CancellationToken));
```

```
public async Task LoginAsync(string username,string password,CancellationToken
    cancellation=default(CancellationToken));
```

```
public async Task<AuthenticatedUser> GetCurrentUserAsync(CancellationToken cancellation=default(CancellationToken));
```

```
public async Task LogoutAsync(CancellationToken cancellation=default(CancellationToken));
```

```
public async Task<TagValue[]> GetTagsAsync(CancellationToken cancellation=default(CancellationToken));
```

```
public async Task<TagValue> GetTagValueAsync(Guid id,CancellationToken cancellation=default(CancellationToken));
```

```
public async Task<TagValue> GetTagValueAsync(string projectName,string tagName,CancellationToken
    cancellation=default(CancellationToken));
```

```
public async Task<WriteResult> SetTagValueAsync(string projectName,string tagName,object value,CancellationToken
    cancellation=default(CancellationToken));
```

```
public static T GetValue<T>(TagValue sample);
```

```
public async Task<WriteResult> SetTagValueAsync(Guid id,object value,CancellationToken
    cancellation=default(CancellationToken));
```

```
public async Task SubscribeAsync(Guid[] ids,Func<TagValue[],Task> onChange,CancellationToken cancellation);
```

10. 실행 예제와 연관 문서

JCoreClient_Examples.md와 같은 버전의 examples/*.cs를 함께 보세요. 01/02/03은 각각 완전한 콘솔 프로그램이고 04는 로그인 후 호출하는 보조 함수, 05는 서버에 등록하는 스크립트입니다. API 함수 서명은 본 문서 9절에 있습니다. 서버 스크립트에서 쓰는 JCore.GetTagValue와 외부 SDK의 GetTagValueAsync는 서로 다른 인터페이스입니다.