



RasProLink

Kullanım ve Devreye Alma Kılavuzu

Tek Windows servisi, beş endüstriyel protokol katmanı

Herhangi bir Windows PC'yi gerçek bir EtherCAT, EtherNet/IP ya da PROFINET istasyonuna dönüştürür.

RAS Otomasyon — RasProLink v1.0

Eylül 2026

İçindekiler

Bölüm 1	Giriş	3
Bölüm 2	Sistem Gereksinimleri	5
Bölüm 3	Kurulum.....	6
Bölüm 4	Web Arayüzü.....	7
Bölüm 5	Veri Eşleşmesi ve Etiketler	10
Bölüm 6	EtherCAT Master	13
Bölüm 7	EtherNet/IP Scanner	17
Bölüm 8	EtherNet/IP Adapter	20
Bölüm 9	PROFINET Controller	22
Bölüm 10	PROFINET Device.....	33
Bölüm 11	.NET İstemci Kütüphanesi	36
Bölüm 12	Lisanslama	41
Bölüm 13	Performans ve Gerçek Zamanlı Davranış	43
Bölüm 14	Genel Sorun Giderme.....	44
Ek A	HTTP API Referansı	45
Ek B	Yapılandırma Dosyaları	50
Ek C	Sözlük	52

Bölüm 1

Giriş

1.1 Bu kılavuz kimin için

Bir PC'nin gerçek bir endüstriyel saha yolu protokolünde konuşmasını gerektiren Master, Controller, Scanner, Adapter ya da Device olarak otomasyon mühendisleri ve bu veriyi bir .NET uygulamasından, bir betikten ya da herhangi bir HTTP istemcisinden kullanan geliştiriciler için yazılmıştır.

Seçtiğiniz protokolün temel kavramlarının (istasyon, slave/device, cyclic veri) bilindiği varsayılır; RAS Otomasyon ürünleriyle önceden deneyim sahibi olmak gerekmez.

1.2 RasProLink nedir

RasProLink, PC'nizi beş protokolde gerçek bir endüstriyel haberleşme istasyonuna dönüştüren tek bir Windows servisi: EtherCAT (Master olarak), EtherNet/IP (Scanner/Originator ya da Adapter/Target olarak) ve PROFINET (IO-Controller ya da IO-Device olarak). Aktif protokol katmanı bir web arayüzünden seçilir ve başka donanım, gateway ya da ücretli üçüncü taraf yığın gerekmez.

Protokol Katmanı	Protokol	Rol
EtherCAT Master	EtherCAT	Master (PC'niz slave cihazları kullanır)
EtherNet/IP Scanner	EtherNet/IP (CIP)	Originator — Adapter'lara Class 1 bağlantı açar
EtherNet/IP Adapter	EtherNet/IP (CIP)	Target — bir Scanner/PLC'den Forward Open kabul eder
PROFINET Controller	PROFINET	IO-Controller — Profinet Master
PROFINET Device	PROFINET	IO-Device — gerçek bir PLC'ye PC, GSDML ile tanımlanır

Aynı anda yalnızca bir protokol katmanı çalışır; web arayüzünden protokol katmanını değiştirmek, alttaki servisi saniyeler içinde yeni protokolde yeniden başlatır. Her protokol katmanı kendi yapılandırmasını korur, bu yüzden geri döndüğünüzde kaldığınız yerden devam eder.

1.3 Neden RasProLink

- Ekstra donanım gerektirmez, dongle ya da gateway cihazları gerektirmeden standart bir Ethernet portu tüm işlevi üstlenir.
- Pahalı 3. Parti ya da bağımsız bağlantı kütüphanelerine gerek yok — protokol yığını dahili ve doğrudan sürülmektedir.
- Tasarım gereği çoklu protokol dizaynı ile yazılım güncellemesi yapılmadan müşteri beklentilerine göre protokol anında değiştirilebilir.
- Sade bir Windows ortamında hiçbir ekstra pakete ihtiyaç duymadan birkaç basit işlemle kolayca devreye alınabilir.

- .NET mimarisi — Mevcut .NET yazılımınız içerisine %100 uyum sağlar. Protokol yığınları ya da endüstriyel haberleşme standartlarını bilmeniz gerekmez. Basit birkaç satırlık kod ile endüstriyel haberleşme yeteneği kazandırabilirsiniz.

İpucu: Gerçek zamanlı not: beş protokol katmanının hepsinde cyclic alışveriş döngüsü; Windows multimedia-timer çözünürlüğü ve CPU çekirdeği sabitlemesiyle özel, yüksek öncelikli bir iş parçacığı üzerinde çalışır; çevrim başına heap tahsisi ortadan kaldırılmıştır. Tipik bir Windows PC'de bu, çevrim jitter'ını milisaniyenin altında tutar (4 ms'lik bir EtherCAT çevriminde ölçülen değerler 0.3 ms civarındadır) — bkz. §13.4.

1.4 Kullanım yolları

Cihazlar bağlandıktan sonra veriyi kullanmanın iki yolu vardır:

- Web arayüzü — cihaz ekleme, durumları izleme, ham I/O baytlarını okuma/yazma, lisans yönetimi ve servis günlüğünü takip. Devreye alma ve tanılama içindir, üretim mantığını çalıştırmak için kullanıcının kendi uygulamanız tarafından yürütülür.
- .NET istemci kütüphanesi (RasProLink.Client) — kendi uygulamanızın cihaz durumunu ve IO'yu kodda okumak, çıkış yazmak ve cihaz/etiketleri programlama ve yönetmek için referans aldığı, bağımlılıksız küçük bir DLL'dir. Gerçek bir uygulama bunun üzerine kurulur — bkz. Bölüm 11.

Bölüm 2

Sistem Gereksinimleri

- Windows 10 / 11 ya da Windows Server 2016 veya üzeri, 64-bit.
- Saha cihazlarıyla aynı fiziksel/mantıksal ağda, boşta kablolu bir Ethernet kartı. Beş protokol katmanının hiçbiri Wi-Fi ile uyum sağlamaz. Gerçek zamanlı cyclic haberleşme trafiği kablolu bir bağlantı ister.
- Yalnızca EtherCAT Master protokolü için dahili ham veri sürücüsünün desteklediği bir yonga sete sahip ağ kartı (çoğu Intel ve Realtek kartı desteklenir) gereklidir.
- PROFINET Controller ve PROFINET Device: ham veri erişimi için Npcap (<https://npcap.com>) — kurulum paketiyle birlikte gelir ya da önceden kurulabilir.
- Servisi çalıştırmak için .NET çalışma zamanı ayrıca kurulması gerekmez — kendi içinde tam bağımsız yayınlanmıştır.
- Windows servisini ve Npcap/ham veri sürücüsünü kurmak için Yönetici hakları gerekir.

Uyarı: Hiçbir protokol katmanının saha tarafını bir Wi-Fi bağlantısının, VPN'in ya da ağır yüklü paylaşılan bir switch üzerine koymayın: EtherCAT ve PROFINET cyclic veri ile EtherNet/IP CIP bağlantıları zamanlamaya duyarlıdır; eklenen gecikme ya da jitter doğrudan bağlantı kaybı ya da watchdog zaman aşımı olarak karşınıza çıkar. Veri yolunun stabilitesi sistem performansını etkileyebilir. Bu durumda birçok cihazın haberleştiği ağır ağ trafiğinden uzak durmalısınız. Yalnızca cihaz haberleşmelerinin aktif olduğu ağlar tercih sebebidir.

Bölüm 3

Kurulum

3.1 Kurulum adımları

1. Paketi hedef PC'ye kopyalayın.
2. Bir npcap-*.exe kurulum dosyası dahilse önce onu çalıştırın (ya da npcap.com'dan önceden Npcap kurun). EtherNet/IP protokol katmanları için gerekmez.
3. install.bat dosyasına sağ tıklayıp Yönetici olarak çalıştır deyin.
4. Servis kurulur, otomatik başlar ve Windows ile birlikte başlayacak şekilde ayarlanır.
5. Web arayüzünün yanıt verdiğini doğrulamak için o PC'de bir tarayıcıda <http://localhost:8080> açın.

3.2 Servis nasıl çalışır

RasProLink, penceresi ya da sistem tepsisi simgesi olmayan, RasProLink.Service adıyla bir Windows servisi olarak çalışır. Kimse oturum açmadan önce başlar ve kullanıcı oturumları arasında çalışmaya devam eder. Durumu kontrol etmek ya da sorun giderirken elle durdurup başlatmak için Windows Hizmetler konsolunu (services.msc) kullanın.

3.3 Web arayüzüne ağdan erişim

Varsayılan olarak web arayüzü tüm yerel adreslerde 8080 portunu dinler, bu yüzden <http://<pc-ip>:8080> ağdaki başka bir makineden ona ulaşır — endüstriyel PC başka bir yere monte edilmişken bir dizüstü bilgisayardan devreye almak için kullanışlıdır. Windows Güvenlik Duvarı etkinse, servise ilk kez dışarıdan erişildiğinde gelen TCP 8080 trafiğine izin verin.

3.4 Güncelleme ve kaldırma

Güncellemek için yeni sürümün install.bat'ını mevcut kurulumun üzerine çalıştırın: çalışan servisi durdurur, yürütülebilir dosyayı ve dosyaları değiştirir, yeniden başlatır. Yapılandırma dosyalarına bir güncellemede dokunulmaz, bu yüzden cihaz listeleri ve ağ ayarları korunur. Kaldırmak için uninstall.bat'ı çalıştırın; yapılandırma klasörü varsayılan olarak korunur (daha sonra yeniden kurarsanız diye) — onu ve günlükleri de silmek için - RemoveConfig geçirin.

Bölüm 4

Web Arayüzü

Web arayüzü yerleşimi beş protokol katmanında da aynıdır: yalnızca aktif protokol katmanının adı ve protokole özgü durum kartları değişir. Aşağıdaki ekran görüntüleri temsili örnek olarak PROFINET Controller protokol katmanından alınmıştır; EtherCAT Master, EtherNet/IP Scanner/Adapter ve PROFINET Device aynı çerçeveyi kendi cihaz/yapılandırma durum kartlarıyla kullanır (kendi bölümlerinde, §6--§10'da ele alınır).

4.1 Üst çubuk ve dil seçimi



Üst çubuk — aktif protokol katmanı adı, dil seçici, canlı bağlantı göstergesi (örnek: PROFINET Controller)

Üst çubuk ürün adını ve aktif protokol katmanını, bir dil seçiciyi ve tarayıcı servise ulaşabildiğinde yeşile dönen bir bağlantı durumu ibaresini gösterir. Tüm arayüz bir dil değişikliğinde anında yeniden uygulanır; sayfa yeniden yüklenmez.

4.2 Lisans



Deneme süresi boyunca Lisans kartı

Bir anahtar etkinleştirildikten sonra Lisans kartı

Kart; durumu, müşteriye, lisans numarasını ve kalan deneme gününü; lisans talep ederken RAS Otomasyon ile paylaşılacak Makine Kimliği'ni; ve bir lisans anahtarı yapıştırma alanını gösterir. Tam akış Bölüm 12'de.

4.3 Ağ kartı seçimi

Ağ kartı seçimi, isteğe bağlı elle MAC geçersiz kılma ile

Birden çok ağ kartı olan bir PC'de, saha cihazlarına fiziksel olarak kablolanmış olanı burada seçin. Bu seçici tüm protokol katmanlarında paylaşılır, çünkü her protokol katmanı sahayla aynı anda tam olarak bir adaptör üzerinden konuşur.

4.4 Aktif protokol katmanını değiştirme

Her protokol katmanının üst çubuğu (§4.1), ortak bir protokol seçim ekranına daima açık bir bağlantı taşır — örneğin PROFINET Controller sayfalarında Protokol: PROFINET Controller ⇄, EtherCAT Master sayfalarında Protokol: EtherCAT Master ⇄ şeklinde görünür. O an hangi protokol katmanı aktif olursa olsun, bu bağlantı her zaman /engine.html adresinde sunulan aynı sayfayı açar ve servisin beş protokol katmanından hangisini çalıştıracağını değiştirmenin tek yoludur.

Seçim ekranı

Seçim ekranı her protokol katmanı için bir kart gösterir — EtherCAT Master, EtherNet/IP Scanner, EtherNet/IP Adapter, PROFINET Controller, PROFINET Device. O an aktif olan protokol katmanının kartında bir AKTİF rozeti ve devre dışı bırakılmış bir Şu an aktif butonu bulunur; diğer her kart ya bir Kullanıma hazır rozeti, ya da henüz tamamlanmamış bir protokol katmanı için kısa bir uyarı satırıyla birlikte bir Geliştirme aşamasında rozeti — ve bir

Bu protokolü seç butonu gösterir. Bu kılavuzun yazıldığı tarihte beş protokol katmanının tümü kullanıma hazırdır, dolayısıyla pratikte aktif olmayan her kart şu an Kullanıma hazır rozetini gösterir; Geliştirme aşamasında rozeti ise ileride bu durumda çıkacak bir protokol katmanı için arayüzde zaten hazır bulunur.

Not: Arayüzün geri kalanının aksine (§4.1), bu sayfa dil seçiciden etkilenmez — geldiğiniz protokol katmanı sayfalarında hangi dil aktif olursa olsun etiketleri Türkçe kalır. Yukarıda alıntılanan etiketler (AKTİF, Kullanıma hazır, Bu protokolü seç vb.) RasProLink'in her dil sürümünde göreceğiniz tam metindir.

Protokolü değiştirme

Bir kartın butonuna tıklamak bir onay ister — hedef protokol katmanı henüz kullanıma hazır değilse daha güçlü bir uyarıyla — ve ardından seçilen protokol katmanının anahtarını (örneğin profinet-controller ya da ethernetip-scanner) POST /api/engine ucuna gönderir. Servis bu seçimi Config\enginesettings.json dosyasına yazar ve yaklaşık üç saniye sonra bilerek kapanır. Windows Servisi olarak kurulduğunda (Bölüm 3), RasProLink her kapanışında — bu temiz kapanış dahil — otomatik olarak yeniden başlayacak şekilde yapılandırılmıştır; bu yüzden Servis Denetim Yöneticisi süreci birkaç saniye içinde yeniden çalıştırır.

Not: Bir geçiş, canlı bir devretme değil, gerçek bir süreç yeniden başlatmasıdır: önceki protokol katmanının sayfalarına ya da API'sine hâlâ bakan başka bir sekme ya da istemci, servis kapandığı anda bağlantısını kaybeder ve yeni protokol katmanı başlamayı bitirene kadar cyclic veri durur. Eski bir sekmeyi kullanmaya devam etmek yerine, yeni protokol katmanının çalıştığını seçim ekranının otomatik yönlendirmesiyle (ya da §4.1'deki bağlantı ibaresiyle) doğrulayın. RasProLink kurulu bir servis yerine etkileşimli olarak (dotnet run) çalıştırıldığında, yukarıda anlatılan otomatik yeniden başlama gerçekleşmez — süreç yalnızca kapanır ve elle yeniden başlatılması gerekir.

Yapılandırma protokol katmanı bazında saklanır

Her protokol katmanı, bu kılavuz boyunca kullanılan aynı kuralı izleyerek (§9.7 ve Bölüm 6–10'daki karşılık gelen bölümler) kendi ayarlarını Config\ altında kendi dosyasında/dosyalarında tutar. Protokol katmanı değiştirmek yalnızca Config\enginesettings.json dosyasına yazar — servis, aktif hale gelen protokol katmanı için yapılandırma deposunu kurar ve başka hiçbir protokol katmanının dosyasını açmaz ya da değiştirmez. Bir protokol katmanından uzaklaşıp daha sonra ona geri dönmek, cihaz listesini, hedefleri, etiketleri ve diğer ayarları tam olarak bırakıldığı gibi bırakır.

Protokol Katmanı	Config\ altındaki kendi dosyaları
PROFINET Controller	controllersettings.json, devices.json, tags.json*
PROFINET Device	deviceconfig.json, tags.json*
EtherNet/IP Scanner	ethernetip-scanner-targets.json, ethernetip-scanner-tags.json
EtherNet/IP Adapter	ethernetip-adapter-config.json, ethernetip-adapter-tags.json
EtherCAT Master	ethercat-master-config.json, ethercat-master-tags.json

Not: * Şu anki sürümde doğrulanmış tek istisna: PROFINET Controller ve PROFINET Device, etiket listeleri için aynı Config\tags.json dosyasını okur ve yazar. Tamamen ayrı olan diğer ayarlarının aksine, yalnızca bu iki protokol katmanı arasında geçiş yapmak her birine bağımsız bir etiket listesi vermez. Lisans kartı (§4.2) bundan da etkilenmez — lisansın kendisi beş protokol katmanının tümü tarafından paylaşılan tek bir Config\lisans.json dosyasında tutulur, bu yüzden protokol katmanı değiştirmek asla yeniden etkinleştirme istemez.

Bölüm 5

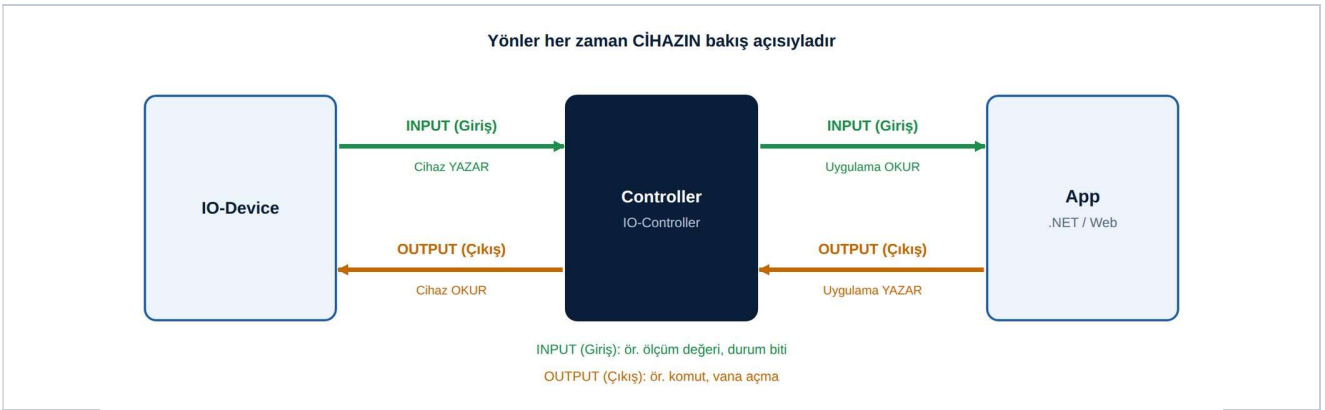
Veri Eşleşmesi ve Etiketler

5.1 Slotlar ve alt modüller

Her protokol katmanı, bağlı bir cihazın process data'sini aynı şekilde düzenler: bir cihaz, her biri bir ya da daha fazla alt modül barındıran ve her alt modül kendi giriş ve/veya çıkış bayt uzunluğunu taşıyan slotlardan oluşan bir ağaçtır. Bu, altta yatan kavram bir PROFINET slot/alt slot, bir EtherCAT PDO-eşlemeli giriş ya da bir EtherNet/IP assembly'si olsa da geçerlidir — RasProLink her üçünü de web arayüzünde, .NET istemcisinde ve HTTP API'de aynı slot/alt slot modeline normalize eder.

5.2 Giriş/çıkış yönü

Yön her zaman bağlı cihazın bakış açısıyla adlandırılır: Giriş, saha cihazının ürettiği ve uygulamanızın okuduğu veridir (bir sensör okuması, bir durum sözcüğü); Çıkış, uygulamanızın ürettiği ve saha cihazının tükettiği veridir (bir setpoint, bir komut, bir bobin). Bu, bir PLC programının kendi I/O'sunu adlandırmasının tersidir ve protokol katmanı başına değil, tek seferlik bir kuraldır.



Yön her zaman CİHAZIN bakış açısıyla adlandırılır — bu, bir PLC programının tersine bir kural, bir kez kontrol etmeye değer.

Bir çıkış alt modülünün baytları uygulamanız (ya da protokol katmanına göre saha tarafındaki PLC/Scanner) tarafından yazılır ve bir sonraki çevrimde bağlı taraf tarafından tüketilir.

5.3 Etiketler

Etiketler

Ham veriye (yukarıdaki Canlı IO) ek olarak, bayt/bit konumuna bir AD ve TİP verin — okuma/yazma bundan sonra bu adla yapılabilir. XDB ile eklenen cihazlarda da etiket tanımlanabilir.

AD	SLOT	YÖN	TİP	KONUM	DEĞER	
Sensor_Giris1	2 / 1	Giriş	bool	@0.0	false	<button>Sil</button>
Motor_Calistir	3 / 1	Çıkış	bool	@0.0	true	<button>Yaz</button> <button>Sil</button>

Etiket adı

1/0x0001 — Arayüz (in 0B / out 0B) ▼

Giriş (cihazdan) ▼

Int16 ▼

Ofset
Bit
Uzunluk
Birim

Ekle / Güncelle

Etiketler kartı — bir bayt/bit aralığını isimlendirerek adla okuma/yazma yapın (örnek PROFINET Controller için gösterilmiştir)

Bir etiket, ham cyclic verinin üstünde bir katmandır — veriyi asla kopyalamaz. Bir slot/alt slot içindeki bir bayt/bit konumuna bir ad ve tip verildiğinde, o konum artık ham hex yerine bu adla (uygulamadan ya da kod üzerinden) okunabilir veya yazılabilir. Etiketler herhangi bir protokol katmanındaki herhangi bir cihazda tanımlanabilir — otomatik eklenenler dahil (.xdb, GSDML ya da ESI eşleşmesiyle); modül yapısı yalnızca slot/alt slot düzenini sağlar, isimlendirme her zaman sonradan eklenen bir katmandır.

Tip	Boyut	Notlar
Bool	1 bit	Bayt içinde 0–7 bit numarası; diğer bitler etkilenmez.
Byte / SByte	1 bayt	İşaretsiz / işaretli 8-bit.
Int16 / UInt16	2 bayt	Büyük-uçlu (big-endian), her üç saha yolunun da hatta kullandığı gibi.
Int32 / UInt32	4 bayt	İşaretli/işaretsiz 32-bit.
Int64 / UInt64	8 bayt	Tam değeri almak için .NET istemcisinde ham baytları okuyun (double 53 bitin üstünü taşıyamaz).
Float32 / Float64	4 / 8 bayt	IEEE 754 (Real / LReal).
String	değişken	Sabit uzunluklu ASCII, sondaki boşluk/NUL ile doldurulmuş kabul edilir.
Bytes	değişken	Ham bayt dizisi, hex metin olarak okunur/yazılır.

Her ekleme/güncellemede kontrol edilen doğrulama kuralları (herhangi bir kural başarısız olursa değişikliğin tamamı reddedilir): ad boş olamaz ve cihaz içinde benzersiz olmalı; bir Bool etiketin bit numarası 0–7 olmalı; bir String/Bytes etiketin uzunluğu en az 1 olmalı; iki etiket aynı slot/alt slot ve yönde aynı baytları kullanamaz (aynı baytın farklı bitlerinde iki Bool etiket bunun tek istisnasıdır — bir durum sözcüğünün her bitini ayrı ayrı isimlendirmenin yaygın bir yolu). Artık var olmayan bir slot/alt slotu işaret eden ya da alt modülünün verisinin

dışına taşan bir etiket, doğrudan engellenmek yerine bir uyarıyla işaretlenir — cihaz farklı bir modül düzeniyle yeniden içe aktarılıncaya kadar zararsızdır.

Bölüm 6

EtherCAT Master

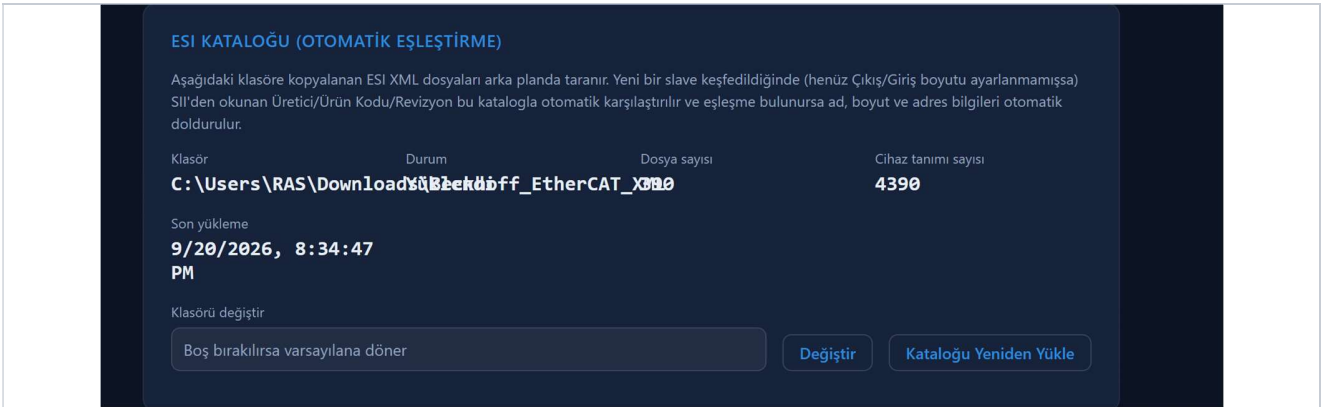
6.1 Genel bakış

EtherCAT Master protokol katmanı, PC'nizin Ethernet kartını bir EtherCAT segmentinin Master'ına dönüştürür: halkayı açar, INIT→PREOP→SAFEOP→OP durum makinesi boyunca yürür ve ardından halkadaki her slave ile yapılandırılmış çevrim süresinde cyclic process-data alışverişini tamamen yazılımda sürür — özel bir EtherCAT master kartı kullanılmaz.

Öge	Değer
Rol	Master
Varsayılan çevrim süresi	4 ms (yapılandırılabilir)
Slave tanımlama	ESI XML kataloğu, SII'den Üretici/Ürün Kodu/Revizyon ile otomatik eşleştirilir
Ölçülen jitter (tipik PC, 4 ms çevrim)	Gerçek zamanlı ince ayardan sonra ≈ 0.3 ms

6.2 ESI Kataloğu — otomatik slave eşleştirme

EtherCAT slave'leri, PROFINET cihazlarının aksine bir vekil dosyaya ihtiyaç duymaz: her slave, gücü verilip halkaya kablolandığı anda kendi Üretici Kimliği'ni, Ürün Kodu'nu ve Revizyon'unu SII (Slave Information Interface) üzerinden kendisi yayınlar. RasProLink bu kimliği otomatik okur ve bir ESI XML dosyaları klasöründe arar — slave üreticilerinizin zaten TwinCAT, Codesys ya da benzeri araçlar için yayınladığı aynı dosyalar.



ESI Kataloğu kartı — ESI dosyaları için izlenen klasör, indekslenen cihaz tanımları, son yeniden yükleme zamanı

Slave üreticilerinizin ESI XML dosyalarını izlenen klasöre kopyalayın (varsayılan bir konum sağlanır ve bu karttan değiştirilebilir); yeni dosyalar otomatik algılanır. Yeni keşfedilen bir slave'in henüz giriş/çıkış boyutu yapılandırılmamışsa, RasProLink SII'den raporlanan imzasını indekslenen her ESI dosyasıyla karşılaştırır ve bir eşleşme bulunduğunda adını, PDO boyutlarını ve process-data ofsetlerini otomatik doldurur — elle bayt saymaya gerek yoktur.

6.3 Process-data eşleşmesi

Her slave'in PDO (Process Data Object) haritası — giriş ve çıkış bit/bayt düzeni — eşleşen ESI girişinden gelir; bu, tıpkı bir PROFINET cihazının Slot/Alt Slot tablosu gibi işler (§5'daki ilkeler burada da geçerlidir: yön her zaman slave'in bakış açısıyla adlandırılır, etiketler bir bayt/bit aralığını isimlendirmenizi sağlar, Bool'dan Bytes'a tüm desteklenen tipler mevcuttur). Beş protokol katmanının tümünün kullandığı paylaşılan veri eşleşme ve etiket kuralları için §5'ya bakın.

6.4 .NET istemci örneği

```
using RasProLink.Client;

using var client = new RasProLinkClient("localhost", 8080);
var status = await client.ConnectAsync();

var slaves = await client.GetDevicesAsync();
foreach (var s in slaves)
    Console.WriteLine($"{s.Name} @ {s.State} in={s.InputBytes}B out={s.OutputBytes}B");

// Read one slave's live process data
var io = await client.GetDeviceIoAsync("drive-axis-1");

// Write an output byte range (e.g. a drive control word)
await client.WriteDeviceOutputAsync("drive-axis-1", slot: 1, subslot: 1,
    data: new byte[] { 0x06, 0x00 });
```

Bu, her protokol katmanının kullandığı aynı RasProLink.Client kütüphanesidir — yalnızca hedef cihaz adı değişir. Tam API yüzeyi için Bölüm 11'a bakın.

6.5 Sorun giderme

Belirti	Olası neden
Hiç slave bulunamadı	Halka kapalı değil (son slave'in OUT portu geri dönmeli, ya da açık halka topolojisi tutarlı kullanılmalı), ya da seçili adaptör halkaya kablolanmış olan değil.
Slave OP altında takılı kalıyor	ESI eşleşmesi yok (bilinmeyen PDO boyutları) — üreticinin ESI dosyasını katalog klasörüne ekleyip yeniden taratmasını sağlayın, ya da slave'e özgü bir SDO/mailbox başlangıç parametresi başarısız oldu (nesneyi SDO aracıyla elle test edin, §6.7).
Yüksek jitter / ara sıra çerçeve kaybı	Aynı CPU çekirdeği için rekabet eden başka bir uygulama, ya da PC'nin güç planının kısması — §1.3 ve §13.4'teki gerçek zamanlı notlara bakın.

6.6 ESI dosyası önizleme ve zincir oluşturucu — kablolamadan önce topolojiyi yapılandırma

§6.2, otomatik eşleştirmeyi anlatır: gücü verilip halkaya kablolanmış bir slave, kimliğini SII üzerinden bildirir, RasProLink de bunu izlenen ESI klasöründe arar. Slave Ayrıntısı kartındaki ESI Dosyası Önizleme ise tam tersi yönde çalışır — tek bir ESI XML dosyası elle seçilir, RasProLink onu tarayıcıda ayrıştırır ve Sync Manager boyutlarını/adreslerini ve CoE desteğini o anda seçili slave'in formuna aktarabilir; ya da henüz halkada hiçbir donanım yokken tüm bir slave dizisini kurmak için de kullanılabilir.

Tek bir ESI dosyasını önizleme

Önce düzenlenecek slave'i slave tablosundan seçin, ardından ESI Dosyası Önizleme altından bir ESI XML dosyası seçin. Dosya birden çok cihaz tanımlıyorsa Cihaz açılır menüsünden doğru olanı seçin; kart ardından Üretici, Ürün Kodu / Sürüm, CoE (Mailbox) desteklenip desteklenmediğini ve dosyanın bildirdiği Çıkışlar (Outputs SM) / Girişler (Inputs SM) boyutlarını gösterir. Bu değerleri forma aktar düğmesine tıklandığında, seçili slave'in çıkış/giriş boyutu ve fiziksel adres alanları o ESI girişinden doldurulur — değerleri elle yazdığınızda olduğu gibi (§6.3); ardından yine formdaki Kaydet'e tıklamanız gerekir.

Bir zincir oluşturma: coupler + terminaller

EtherCAT'te modüler IO genelde bir bus coupler'dır (ör. Beckhoff EK1100) ve üzerine bir dizi terminal takılır (ör. EL1008, EL2008); her terminal halkada kendi başına bir slave'tir ve bir önceki slave'in hemen ardından gelen pozisyonda yer alır. Bunların her birini tek tek ESI önizleyip kaydetmek yerine, coupler'ın ESI dosyasını yükleyip Zincire ekle'ye tıklayın, ardından fiziksel, soldan sağa kablolama sırasına göre her terminal için bunu tekrarlayın — her Zincire ekle, o anda önizlenen cihazı aşağıdaki Zincir (Coupler + Terminaller) tablosuna ekler; burada pozisyonu, Ürün Kodu / Sürümü, adı, çıkış/giriş bayt sayıları ve CoE desteği listelenir; bir satır uygulanmadan önce Yukarı/Aşağı taşı ile yeniden sıralanabilir ya da Kaldır ile atılabilir.

Başlangıç halka pozisyonunu ayarlayın (boş bir zincire ilk eklemede, listedeki son slave'den bir sonraki pozisyona varsayılan olarak ayarlanır) ve Zinciri slave listesine uygula'ya tıklayın: RasProLink zinciri sırayla dolaşır, her girişe başlangıç pozisyonu + indeks halka pozisyonunu ve 0x1000 + halka pozisyonu sabit istasyon adresini atar, ardından bunu normal bir PUT ile /api/ethercat/slaves'e kaydeder — elle doldurulan formun kullandığı aynı çağrı. Eğer o halka pozisyonunda zaten bir slave varsa, kopyalanmak yerine mevcut giriş yerinde güncellenir, böylece düzeltilmiş bir zinciri mevcut olanın üzerine yeniden uygulamak güvenlidir. Sonuçta ortaya çıkan, elle girilenlerden ya da ESI kataloğuyla eşleştirilenlerden (§6.2) ayırt edilemeyen sıradan bir slave yapılandırma girişleri kümesidir — zincir tablosunun kendisi yalnızca bir hazırlama alanıdır ve uygulandıktan sonra temizlenir.

İpucu: Bu sayede pano henüz kablolanmadan, üreticinin datasheet'indeki ESI dosyalarından tüm bir coupler + terminal dizisini masanızda yapılandırabilirsiniz — bir makineyle birlikte gönderilecek yapılandırma dosyasını önceden hazırlamak için kullanışlıdır. Gerçek donanım kablolanıp veriyolu açıldığında, her slave'in gerçek SII kimliği, yapılandırılmış Üretici Kimliği/Ürün Kodu/Sürümüne karşı diğer her slave'de olduğu gibi kontrol edilir (§6.1); bir uyumsuzluk (yanlış slotta yanlış terminal) slave tablosundaki ESM Durumu sütununda hemen görülür.

6.7 SDO (CoE) okuma/yazma — elle CANopen-over-EtherCAT erişimi

§6.3'te anlatılan döngüsel PDO alışverişinin yanı sıra, CoE (CANopen over EtherCAT) destekleyen bir slave, nesne sözlüğünü mailbox üzerinden de sunar — 16 bit indeks ve 8 bit alt indeksle adreslenen tek tek nesneler, döngüsel çerçeveden bağımsız olarak SDO (Service Data Object) aktarımlarıyla teker teker okunur ya da yazılır. Bu, elle yapılan isteğe bağlı bir yoldur: her döngüde çalışmaz ve devreye alma/tanımlama içindir, kontrol döngüsü verisi için değildir (her döngüde taşınması gereken her şey için etiketleri/PDO eşleşmesini kullanın, §6.3).

Slave tablosunda bir slave seçin; SDO (CoE) Okuma/Yazma kartı, Slave Ayrıntısı kartında Canlı IO'nun (§6.9) altında, yalnızca o slave'in ESI'de bildirilen CoE desteği ayarlıysa görünür — aksi halde kart Bu slave CoE (mailbox) desteklemiyor yazısını gösterir ve alanlar gizli kalır. İndeks'i (hex, ör. Identity Object için 1018) ve Alt İndeks'i girin, ardından mevcut değeri hex olarak almak için Oku'ya tıklayın, ya da Veri (hex) yazıp göndermek için Yaz'a tıklayın. İkisi de döngüsel PDO çerçevesi yerine slave'in mailbox'ından geçer, bu yüzden bir okuma/yazma bir veriyolu döngüsünden belirgin şekilde daha uzun sürebilir ve henüz PREOP'u geçmemiş bir slave'de başarısız olabilir.

Not: Başarısız bir SDO aktarımı, hata mesajının yanında slave'in CANopen abort kodunu da bildirir — slave'inizin kılavuzunun o nesne için belgelediği aynı abort kodu (ör. nesne mevcut değil, ya da salt okunur bir girişe yazma erişimi reddedildi).

6.8 Ağ kartı seçimi ve veriyolu çevrim süresi

§6.1, varsayılan 4 ms çevrim süresini motorla ilgili bir olgu olarak belirtir; bu bölüm, hem ağ kartının hem de o çevrim süresinin fiilen ayarlandığı yerdire — web arayüzünün üst kısmındaki Ağ Kartı Seçimi kartı.

Ağ Kartı Seçimi

EtherCAT, IP yerine ham Ethernet çerçeveleriyle çalışır, bu yüzden Master'a halkanın fiziksel olarak hangi ağ kartına bağlı olduğu söylenmelidir. Ağ Kartı açılır menüsü RasProLink'in görebildiği her adaptörü listeler; Master'ın seçmesine izin vermek için (otomatik seç) üzerinde bırakın, ya da açıkça birini seçin. MAC (elle, isteğe bağlı) alanı, adaptörü işletim sisteminin atadığı ad yerine MAC adresiyle sabitlemenizi sağlar; bu, sürücü güncellemesinden sonra ya da bir USB dock yeniden bağlandığında adaptör adlarının değişebildiği bir PC'de kullanışlıdır. Uygula'ya tıklamak veriyolunu yeniden başlatır — bir NIC değişikliği tüm segmenti etkiler, bu yüzden tıpkı bir slave ekleme/çıkarma (§6.3) gibi halkanın kısa süreliğine yeniden açılmadan uygulanamaz.

Veriyolu çevrim süresi

Aynı karttaki Döngü Aralığı (ms) alanı, Master'ın döngüsel alışverişi ne sıklıkla sürdüğünü ayarlar (§6.1'deki 4 ms yalnızca bu alanın varsayılanıdır, sabit bir değer değildir). Yeni bir aralık girin ve yanındaki Uygula'ya tıklayın: adaptörden farklı olarak, çevrim süresini değiştirmek halkanın FMMU/Sync Manager yapılandırmasını yeniden kurmaz, bu yüzden yeniden başlatmayı zorlamaz — arayüzün kendi ifadesi açıktır: Uygulandı, yeni değer veriyolu bir sonraki açılışında devreye girer; yani akış ortasında değil, bir sonraki bağlantı döngüsünde alınır.

Not: "Veriyolu bir sonraki açılışında" uygulamada bir servis yeniden başlatmasından daha erken gerçekleşir: durum kartından (§6.1) Master'ı durdurup başlatmak, ya da zaten bir yeniden başlatmayı tetikleyen herhangi bir değişiklik (yukarıdaki bir NIC değişikliği, ya da bir slave ekleme/çıkarma, §6.3) da halkayı yeniden açar ve yeni çevrim süresini devreye alır.

6.9 Canlı IO — web arayüzünde slave başına elle okuma/yazma

§6.4, process data'nın .NET istemci kodundan nasıl okunup yazılacağını gösterir; aynı ham baytlara, tek satır kod yazmadan, doğrudan tarayıcıdan da Slave Ayrıntısı kartındaki Canlı IO paneli üzerinden ulaşılabilir.

Slave Ayrıntısı kartını açmak için slave tablosunda bir slave seçin; kimlik alanlarının (ESM Durumu, Vendor ID, Ürün Kodu, Sürüm, Seri No) altında yer alan Canlı IO bloğu iki alan gösterir: Giriş (slave'den gelen, salt okunur, hex) slave'in geçerli giriş görüntüsüdür, otomatik yenilenir ve salt okunurdur; Çıkış (slave'e giden, hex) ise düzenlenebilir bir alandır — bir hex dizesi yazıp Gönder'e tıklayarak bir sonraki döngüde onu doğrudan o slave'in çıkış görüntüsüne yazabilirsiniz. Bu, istemci kodunun GetDeviceIoAsync/WriteDeviceOutputAsync (§6.4) ile eriştiği aynı giriş/çıkış bayt aralığıdır; bir istemci projesi olmadan tezgah testleri için doğrudan sunulur — örneğin bir pano kablolanırken bir çıkış bobinini zorlamak, ya da bir sensör tetiklendiğinde, daha etiket adları bile tanımlanmadan (§6.3), bir giriş baytının değiştiğini doğrulamak için.

Not: Bu şekilde gönderilen bir değer, bağlı bir istemcinin bir sonraki döngüsel yazımı ya da bir etiket yazımı (§6.3) ile üzerine yazılır — Canlı IO bir çıkışı tutmaz ya da kilitlemez, yalnızca devam eden döngüsel alışverişe tek bir yazma enjekte eder. Bu, tezgah testi ve devreye alma içindir, çalışan bir istemci uygulamasının yerini tutmaz.

Bölüm 7

EtherNet/IP Scanner

7.1 Genel bakış

EtherNet/IP Scanner protokol katmanı, PC'nizi bir ya da daha fazla Class 1 (implicit, cyclic) CIP bağlantısının Originator'u yapar: yapılandırılan her Target/Adapter'a bir Forward Open açar, ardından müzakere edilen RPI'de (Requested Packet Interval) giriş ve çıkış assembly'lerini alışveriş eder ve bağlantı düşerse otomatik olarak yeniden açar.

Öge	Değer
Rol	Originator
Bağlantı sınıfı	CIP Class 1 (cyclic, UDP), Target başına bir Forward Open
Tanımlama	Target IP + assembly örnek numaraları (giriş/çıkış/yapılandırma)

7.2 Bir Target ekleme

Web arayüzünün cihaz kartından IP adresini, EDS dosyasından ya da veri sayfasından alacağınız assembly örnek numaralarını (giriş, çıkış ve kullanılıyorsa yapılandırma assembly'si) ve istenen RPI'yi girerek bir Target ekleyin — ya da önce bir Cihaz Keşfi taraması çalıştırarak (§7.5) elle aramayı tamamen atlayın; bu tarama aynı formu önceden doldurabilir. RasProLink Forward Open'ı hemen açar ve ilk cyclic alışveriş başarılı olduğunda Target'ın satırı Bağlanıyor → Çalışıyor gösterir — PROFINET Controller'ın cihaz listesinde kullanılanla (§9.2) aynı durum akışı.

7.3 .NET istemci örneği

```
using RasProLink.Client;

using var client = new RasProLinkClient("localhost", 8080);
await client.ConnectAsync();

var targets = await client.GetDevicesAsync();
foreach (var t in targets)
    Console.WriteLine($"{t.Name} {t.IpAddress} [{t.State}] in={t.InputBytes}B out={t.OutputBytes}B");

var io = await client.GetDeviceIoAsync("vfd-conveyor-3");
await client.WriteDeviceOutputAsync("vfd-conveyor-3", slot: 0, subslot: 1,
    data: new byte[] { 0x01, 0x00 }); // e.g. run/stop word
```

7.4 Sorun giderme

Belirti	Olası neden
Forward Open reddedildi	Assembly örnek numaraları ya da boyutları Target'ın EDS'iyle eşleşmiyor, ya da Target'ta zaten başka bir Scanner/PLC'den açılmış azami sayıda bağlantı var.
Bağlantı tekrar tekrar zaman aşımına uğruyor	RPI, ağ yolu için çok agresif ayarlanmış, ya da bir anahtar/VLAN gecikme ekliyor — §2'deki gerçek zamanlı ağ rehberliğine bakın.

7.5 Cihaz Keşfi — hattaki Target'ları bulma

Cihaz Keşfi kartı bir UDP ListIdentity broadcast'i gönderir ve yanıt veren her EtherNet/IP cihazını IP adresi, Vendor ID, Ürün Kodu, sürüm ve ürün adıyla listeler — PROFINET Controller'ın DCP ağ taramasının (§9.3) EtherNet/IP karşılığıdır. Sabit bir süre boyunca çalışır (varsayılan 2 sn) ve kendi başına hiçbir CIP bağlantısı açmaz; salt okunur bir keşiftir.

Her sonuç bir EDS Eşleşmesi sütunu taşır: RasProLink, yanıt veren cihazın Vendor ID/Ürün Kodu/Cihaz Tipi/sürümünü EDS Kataloğu'na (§7.6) karşı otomatik olarak arar. "Bu IP'yi kullan" düğmesi, Target Ekleme formunun (§7.2) yalnızca IP alanını doldurur. Bir katalog eşleşmesi bulunduğunda ise bunun yerine "Otomatik Doldur" sunulur; bu seçenek, eşleşen EDS'ten assembly örnek numaralarını ve giriş/çıkış boyutlarını da önceden doldurarak Target'ın hiçbir şeye elle bakılmadan eklenmesini sağlar.

Not: ListIdentity, yerel alt ağda bir broadcast'tir — bir router ya da VLAN sınırının arkasındaki bir Target, bilinen IP'sine bir Forward Open yine de başarılı olsa bile burada görünmez. Bu durumda cihazı elle ekleyin.

7.6 EDS Kataloğu — otomatik eşleştirme

EDS Kataloğu kartı, üretici EDS dosyalarından oluşan bir klasörü izler ve arka planda tarar — PROFINET GSDML kataloğunun (§9.10) EtherNet/IP karşılığıdır. Cihaz üreticilerinizin sağladığı .eds dosyalarını klasöre bırakın (varsayılan klasör, RasProLink'in diğer yapılandırma dosyalarının yanındaki ethernetip-scanner-eds'tir) ve RasProLink her cihaz tanımının Vendor ID/Ürün Kodu/Cihaz Tipi/sürümünü assembly düzeniyle birlikte indeksler; böylece Cihaz Keşfi (§7.5) tek bir EDS dosyası bile elle açmadan eşleştirme ve ön doldurma yapabilir.

Alan	Anlamı
Klasör	İzlenen dizin; Değiştir/Gözet denetimleriyle değiştirin (RasProLink etkileşimli bir masaüstünde çalışıyorsa bir klasör seçici sunulur, aksi halde yolu doğrudan yazın).
Durum / Dosya sayısı / Cihaz tanımı sayısı	Son taramanın yüklenip yüklenmediği ya da hâlâ çalışıp çalışmadığı, kaç .eds dosyasının okunduğu ve bunlardan kaç cihaz tanımının çıkarıldığı (bir dosya birden çok cihaz varyantını tanımlayabilir).
Kataloğu Yeniden Yükle	Klasörün arka plan izleyicisini beklemeden hemen yeniden taranmasını zorlar — yeni EDS dosyalarını kopyaladıktan hemen sonra kullanın.

Not: Ayırıştırılamayan bir dosya atlanır; bu ölümcül bir hata değildir, dosya yalnızca kataloğun hata listesinde adıyla bildirilir ve siz onu düzeltebilir ya da değiştirebilirsiniz. Bu sırada klasördeki diğer her dosya normal şekilde yüklenmeye devam eder.

7.7 EDS Dosyası Önizleme — cihaz eklemekten önce dosyayı incelemek

EDS Dosyası Önizleme kartı, tek bir EDS dosyasının ham metnini yapıştırıp EDS Kataloğu klasörüne (§7.6) eklemekten ve bir Target oluşturmadan, olduğu yerde ayrıştırmanıza olanak tanır. Üretici adını ve kodunu, ürün kodunu/tipini/adını, sürümünü ve dosyada tanımlanan her assembly örneğini bayt boyutu ve yönüyle (Giriş/Çıkış) bildirir — bir üreticinin az önce e-postayla gönderdiği bir dosyayı kontrol etmek ya da henüz bağlanmaya hazır olmadığınız bir cihazın assembly numaralarını okumak için kullanışlıdır.

Dosyanın [Connection Manager] yolu sorunsuz çözümlenirse, "Bu değerleri forma aktar" düğmesi görünür ve ayrıştırılan assembly örneklerini ve boyutlarını doğrudan Target Ekleme formuna (§7.2) kopyalar. Yön dosyadan belirlenemiyorsa (kullanılabilir bir Connection Manager yolu yoksa) bu düğme gizlenir ve örnek numaralarını tablodan elle okursunuz.

Bölüm 8

EtherNet/IP Adapter

8.1 Genel bakış

EtherNet/IP Adapter protokol katmanı, PC'nizi bir Target yapar: bir Scanner'dan — tipik olarak bir PLC'den — gelen Forward Open'ı dinler ve kabul edildikten sonra, Scanner'ın istediği RPI'de yapılandırılan giriş/çıkış assembly'lerini alışveriş eder. Bu, Scanner protokol katmanının ayna görüntüsüdür: burada PC'niz taranan cihazdır, tarayıcı taraf değil.

Öge	Değer
Rol	Target / Adapter
Scanner'a tanımlama	Yapılandırılan assembly örnekleri (giriş/çıkış/yapılandırma) ve bayt boyutları

8.2 Yerel assembly'leri yapılandırma

Web arayüzünden, giriş assembly'sini (PC'nizin ürettiği, Scanner tarafından okunan veri) ve çıkış assembly'sini (Scanner'ın yazdığı, PC'niz tarafından tüketilen veri) örnek numaraları ve bayt boyutlarıyla tanımlayın — bağlanan Scanner/PLC'nin kendi yapılandırmasının (ya da EDS'inin) beklediğiyle eşleşecek şekilde. Kaydedildikten sonra, RasProLink o örnek için Forward Open isteklerini kabul etmeye başlar.

8.3 .NET istemci örneği

```
using RasProLink.Client;  
  
using var client = new RasProLinkClient("localhost", 8080);  
await client.ConnectAsync();  
  
// Read what the Scanner/PLC most recently wrote to us  
var io = await client.GetDeviceIoAsync("self");  
  
// Publish new input data for the Scanner/PLC to read next cycle  
await client.WriteDeviceOutputAsync("self", slot: 0, subslot: 1,  
    data: new byte[] { 0x2A, 0x00, 0x00, 0x00 });
```

8.4 Sorun giderme

Belirti	Olası neden
Scanner bağlantı açamıyor	Scanner'ın yapılandırması ile bu protokol katmanının yapılandırılmış assembly'leri arasında örnek/boyut uyumsuzluğu, ya da assembly henüz kaydedilmemiş/başlatılmamış.

8.5 Bağlantı durumu ve istatistikler

Adapter'ın web sayfasındaki iki kart, bir REST çağrısına ya da paket yakalamaya ihtiyaç duymadan, bir Scanner'ın bu protokol katmanıyla tuttuğu bağlantıya canlı görünürlük sağlar — Scanner protokol katmanındaki (§7) gibi ayrı bir cihaz listesi burada yoktur, çünkü Adapter'da her zaman yalnızca bir Scanner'ın açtığı tek bir bağlantı bulunur.

Bağlantı Durumu

Şu anda bir Scanner'ın bağlı olup olmadığını, bağlıysa IP adresini ve bağlantının kurulduğu zamanı ("Bağlantı zamanı") gösterir. Ayrıca Forward Open'ın müzakere ettiği iki CIP bağlantı kimliğini — Input Connection ID (bu protokol katmanının aldığı veriyi taşıyan O→T bağlantısı) ve Output Connection ID (gönderdiği veriyi taşıyan T→O bağlantısı) — ve Scanner'dan alınan son girişin zaman damgasını gösterir. Bu bilgi, bağlantıyı resmi olarak kapatmadan sessizleşen bir Scanner'ı fark etmek için kullanışlıdır.

İstatistikler

Forward_Open Denemesi (bu protokol katmanının işlediği, kabul edilmiş ya da reddedilmiş her Forward Open isteği, mevcut bağlantıdan önceki denemeler dahil), Bağlantı Kaybı (kurulmuş bir bağlantının kaç kez düştüğü), Giden Çerçeve ve Gelen Çerçeve (protokol katmanı başladığından beri cyclic I/O çerçeve sayaçları) ve PROFINET Controller'ın cihaz düzeyindeki jitter grafiğiyle (§9.4, §13.4) aynı şekilde çizilen Jitter'ı bildirir — geniş ya da büyüyen bir jitter okuması, §7.4'ün sorun giderme tablosunda ele alınan aynı ağ yolu nedenlerine işaret eder.

Not: Bu Adapter'a bağlantı açacak şekilde yapılandırılmış olabilecek diğer Scanner/PLC'leri listeleyen bir REST uç noktası ya da kullanıcı arayüzü listesi yoktur — bir Adapter Target her zaman yalnızca o an tuttuğu tek Forward Open'a hizmet eder, dolayısıyla "kim bağlı" sorusunun yanıtı zaten yukarıdaki Bağlantı Durumu kartında gösterilenle aynıdır.

8.6 I/O İzleme / Simülasyon — Scanner olmadan tezgah testi

I/O İzleme / Simülasyon kartı, gerçek bir Scanner ya da saha donanımı bağlı olmadan bu protokol katmanının I/O'sunu çalıştırmanıza olanak tanır. Geçerli giriş assembly'sini salt okunur hex olarak gösterir (bağlı bir Scanner'ın en son yazdığı veri, hiçbiri bağlı değilse tümü sıfır) ve çıkış assembly'si için bir hex alanı ile bir Gönder düğmesi sunar; böylece gerçek bir Scanner'ın aksi halde yazacağı baytları elle enjekte edebilirsiniz — bir PLC programı var olmadan önce etiket tanımlarını (§5) ya da bu protokol katmanına karşı bir entegrasyonu test etmek için kullanışlıdır.

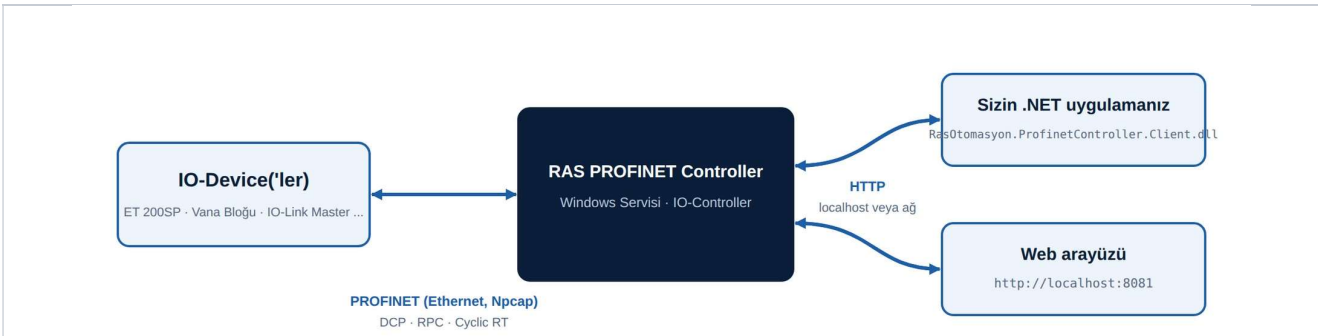
İpucu: Buradaki "Çıkış" ve "Giriş", §8.2'nin alan adlarıyla eşleşecek şekilde Scanner'ın bakış açısını izler: bu kartta yazıp Gönder'e bastığınız veri, bağlı Scanner'ın giriş assembly'si olarak okuduğu veriyle tam olarak aynıdır.

Bölüm 9

PROFINET Controller

9.1 Genel bakış

PROFINET Controller protokol katmanı, Windows PC'nizi normalde bir PLC'nin üstlendiği role sokarak gerçek bir PROFINET IO-Controller'a dönüştürür. Gerçek bir ağ kartı üzerinden, bir PLC'nin kullandığı aynı protokollerle (keşif için DCP, bağlantı kurulumu için RPC, çalışan veri alışverişi için cyclic gerçek zamanlı çerçeveler) bir ya da birden çok 3. parti PROFINET IO-Device'ine doğrudan bağlanır: uzak I/O blokları, vana adaları, IO-Link master'lar, herhangi bir tür GSDML ile tanımlanmış cihaz. Araya bir PLC girmesi hiç gerekmez.

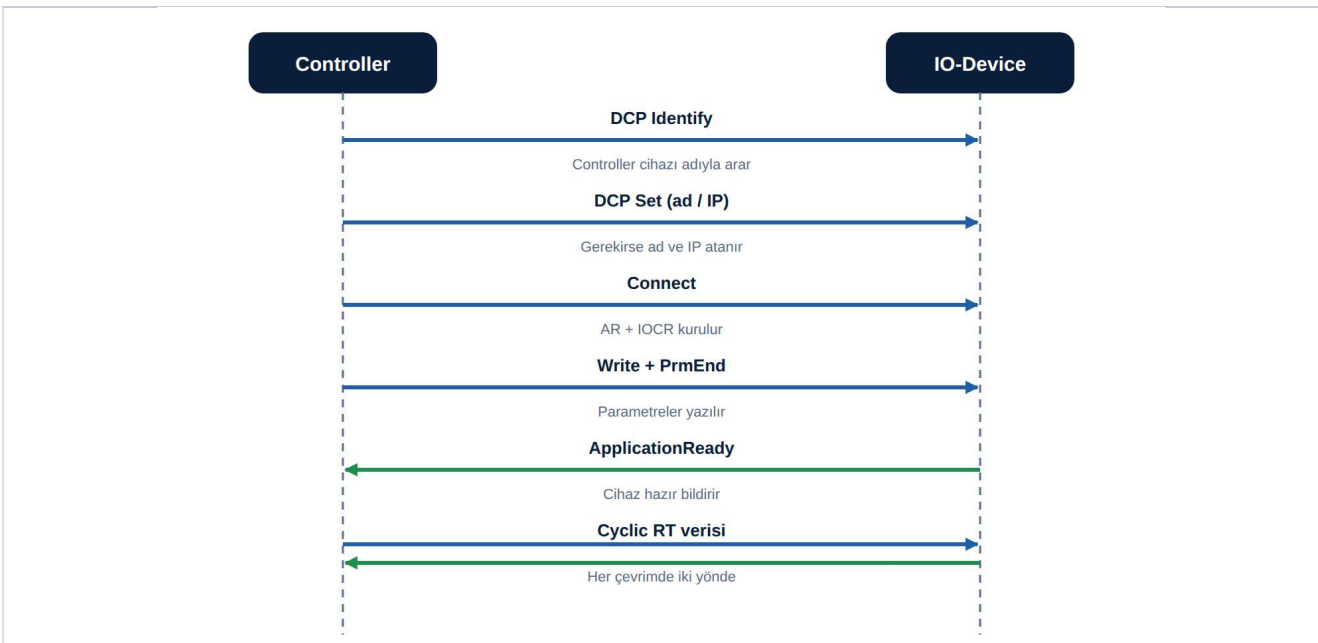


Bir tarafta IO-Device'ler, diğer tarafta uygulamanız ve web arayüzü — Controller servisi ortada durur

Tek bir Controller örneği tek bir cihazla sınırlı değildir: bir ağ kartı ve bir RPC soketi açar, yapılandırılan her IO-Device bunun üzerinden hizmet alır — cihaz sayısı mimari değil, lisans/pratik bir konudur.

9.2 Bir bağlantı nasıl kurulur

Her cihaz aynı sıradan geçer; bu sıra cihaz listesinde durum olarak görünür: Resolving (DCP Identify/Set) → Connecting → Parametrizing → WaitingApplicationReady → Running.



DCP Identify/Set, Connect, Write+PrmEnd, ApplicationReady, ardından her çevrimde iki yönde cyclic RT verisi

9.3 IO-Device'ler ekleme

IO-DEVICE'LER

TIA Portal'dan dışa aktarılan XDB dosyasını buraya yükleyin; içindeki her cihaz otomatik olarak listeye eklenir ve etkinse hemen bağlanmaya çalışılır.

XDB dosyasını buraya sürükleyin ya da seçmek için tıklayın

CİHAZ	IP	DURUM	GİRİŞ / ÇIKIŞ	ÇEVİRİM	ETKİN
ET 200SP Giriş/Çıkış	192.168.0.21	Çalışıyor	1B / 1B	4 ms	☒ Sil
Vana Blok 2	192.168.0.22	Hata	0B / 2B	8 ms	☒ Sil
IO-Link Master 3	192.168.0.23	Kapalı	32B / 32B	16 ms	☐ Sil

IO-Device'ler kartı — içerdği her cihazı tek adımda eklemek için bir .xdb dosyası bırakın

Bir IO-Device eklemenin birbirinden bağımsız iki yolu vardır ve yalnızca biri TIA Portal gerektirir. Ağ Taraması, ağdaki her PROFINET cihazını DCP ile bulur ve her biri için Üretici/Cihaz Kimliğini GSDML kataloğuyla otomatik olarak karşılaştırır — hiç .xdb kullanılmadan (§9.10). Aşağıdaki adımlar ikinci yolu, yani TIA Portal'dan dışa aktarılan bir .xdb dosyasının içe aktarılmasını kapsar.

Bu yol için Controller doğrudan bir GSDML dosyasına ihtiyaç duymaz — bunun yerine, TIA Portal'ın bir projede yapılandırılmış bir cihaz için zaten dışa aktardığı bir cihaz vekil dosyasını (.xdb) okur. Bu dosya, Controller'ın bilmesi gereken her şeyi zaten içerir: cihazın istasyon adı, IP yapılandırması ve her modülün giriş/çıkış bayt uzunlukları ve parametreleriyle birlikte tam slot/alt slot düzeni — hiçbirini elle yeniden girilmez. TIA Portal'ı ve .xdb dosyasını tamamen atlayan alternatif yol için §9.10'a bakın.

Sütun	Anlamı
Cihaz	Görünen ad (yoksa istasyon adına düşer); modül/alt modül uyumsuzluğu olan bir cihazda adının altında küçük bir "Modül uyumsuzluğu" uyarısı da görünür (§9.4).
IP	Cihazın kendi IP adresi.
Durum	Çalışıyor (cyclic veri akıyor), Hata (son deneme başarısız), Kapalı (devre dışı / hiç denenmedi).
Giriş / Çıkış	Cihazın tüm alt modüllerindeki toplam giriş ve çıkış bayt sayısı.
Çevrim	Yapılandırılmış güncelleme süresi.
Etkin	Cihazın yapılandırmasını silmeden bağlanmayı durdurmak için işareti kaldırın.

9.4 Cihaz ayrıntısı, canlı IO ve etiketler

CİHAZ AYRINTISI

Modül Yapısı

SLOT	MODÜL	ALT SLOT	ALT MODÜL	GİRİŞ	ÇIKIŞ	PARAMETRE
1	IM 155-6 PN HF (Kafa Modülü) (0x00000001)	0x0001	Arayüz (0x00000001)	0B	0B	0
2	8DI DC24V (0x0000AB12)	0x0001	8DI DC24V (0x0000AB12)	1B	0B	3
3	8DO DC24V/0.5A (0x0000AB20)	0x0001	8DO DC24V/0.5A (0x0000AB20)	0B	1B	2

Canlı IO

SLOT	MODÜL	VERİ (HEX)	DURUM
2	8DI DC24V	9C	OK
3	8DO DC24V/0.5A	3F Yaz	OK

Etiketler

Ham veriye (yukarıdaki Canlı IO) ek olarak, bayt/bit konumuna bir AD ve TİP verin — okuma/yazma bundan sonra bu adla yapılabilir. XDB ile eklenen cihazlarda da etiket tanımlanabilir.

AD	SLOT	YÖN	TİP	KONUM	DEĞER	
Sensor_Giris1	2 / 1	Giriş	bool	@0.0	false	Sil
Motor_Calistir	3 / 1	Çıkış	bool	@0.0	true Yaz	Sil

İstatistik

Gönderilen 812940	Alınan 812935	Alarm 0	Bağlantı Denemesi 1
Bağlantı Kaybı 0	Jitter 0.280 ms		

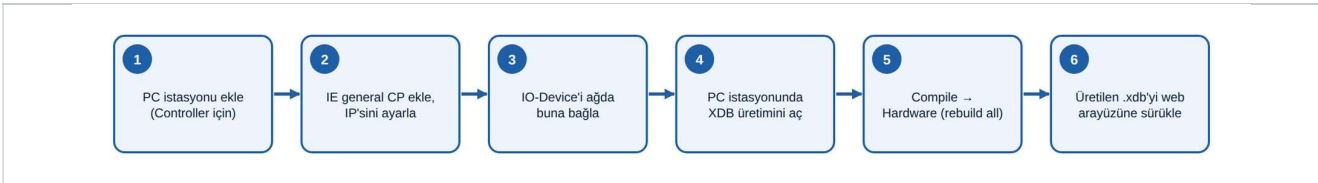
Cihaz ayrıntısı — modül yapısı, elle hex-yazma alanlı canlı IO, etiketler ve jitter dahil bağlantı istatistikleri

Listede bir cihaza tıklamak üç bölüm açar: Modül yapısı (.xdb'den içe aktarılan ya da GSDML kataloğundan yapılandırılan slot/alt slot düzeni, burada salt okunur), Canlı IO (her alt modülün anki ham baytları, hızlı tezgah testleri için çıkış alt modülleri üzerinde bir hex alanı ve Yaz düğmesiyle) ve Etiketler (adıyla ya da ham hex ile değil, bir bayt/bit aralığını isimlendirerek okunmasını/yazılmasını sağlar — kurallar §5'da). Alttaki istatistikler gönderilen/alınan çerçeveleri, alarmları, bağlantı denemelerini, bağlantı kayıplarını ve çevrim jitter'ını (§13.4) raporlar. Jitter için ayrıca sayfanın devamında kendi canlı grafiği vardır; bu grafik, istatistik satırındaki anlık sayının aksine, çalışan min/maks/ortalama izler.

Her Connect'te cihaz, slot/alt slot bazında gerçek modül ve alt modül kimliklerini geri bildirir; Controller bunları yapılandırılmış düzenle karşılaştırır. Hiç modül takılı olmayan, yanlış modül takılı olan ya da eşleşmeyen bir alt modül kimliği olan bir slot, cihazı listede adının altında küçük bir “Modül uyumsuzluğu” uyarısıyla işaretler (§9.3); bazı alt modül durumları (ör. yedek bir değer) veri alışverişini engellemediği için, uyumsuzluk varken de cihaz Çalışıyor durumuna ulaşabilir. Cihaz ayrıntısı görünümünü açmak, cihazın gerçekte bildirdiği her slot ve alt slotu durumuyla (modül düzeyinde modül yok / yanlış modül / doğru modül / yedek modül, ya da alt modül başına ok / yedek / yanlış / yok) birlikte listeleyen bir “Modül farkı” satırı gösterir. Bu satır, panoyu tek tek dolaşmadan hangi fiziksel modülün değiştirildiğini, eksik bırakıldığını ya da yanlış slota kablolandığını bulmak için kullanılır.

9.5 TIA Portal — cihaz vekil dosyasını dışa aktarma

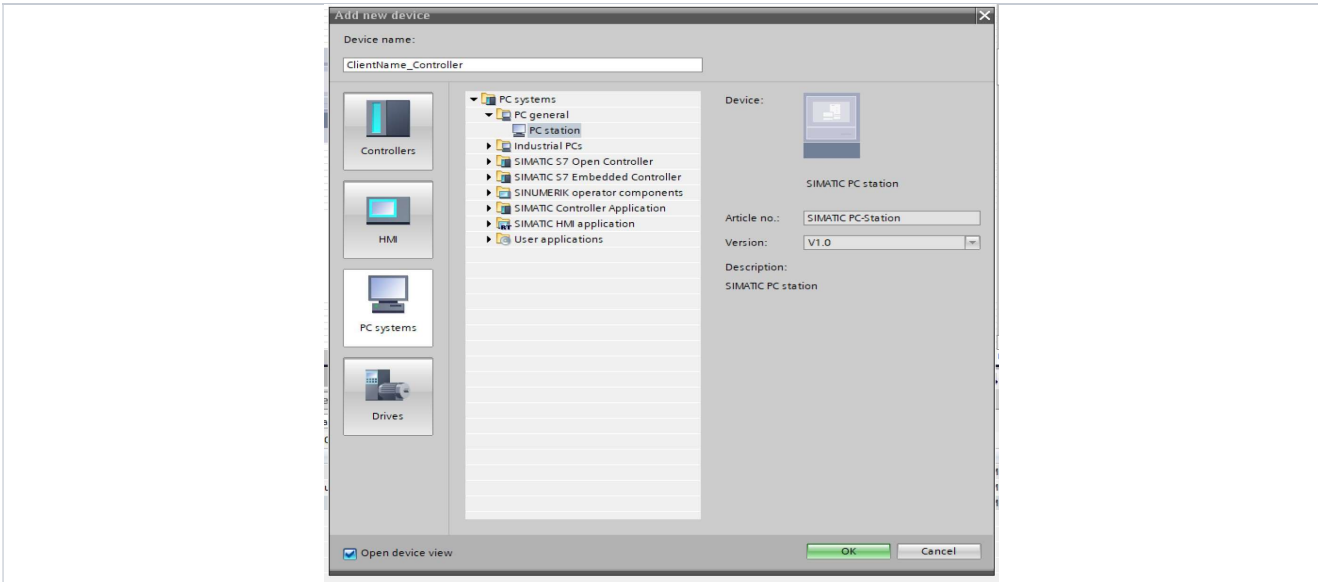
IO-Device zaten gerçek bir PLC projesinde (S7-1200/1500 vb.) yer alıyorsa, ona sağ tıklayıp vekil dosyasını doğrudan dışa aktarın. Ancak çoğu kurulumda fiziksel bir PLC hiç yoktur — o rolü Controller'ın kendisi üstlenir — bu yüzden TIA projesinin onu temsil edecek bir istasyona ihtiyacı vardır. Aşağıdaki adımlar, gerçek bir TIA Portal projesinden ekran görüntüleriyle bu yaygın yolu gösterir.



Genel akış — ayrıntılar aşağıda

1. Controller için bir PC istasyonu ekleyin

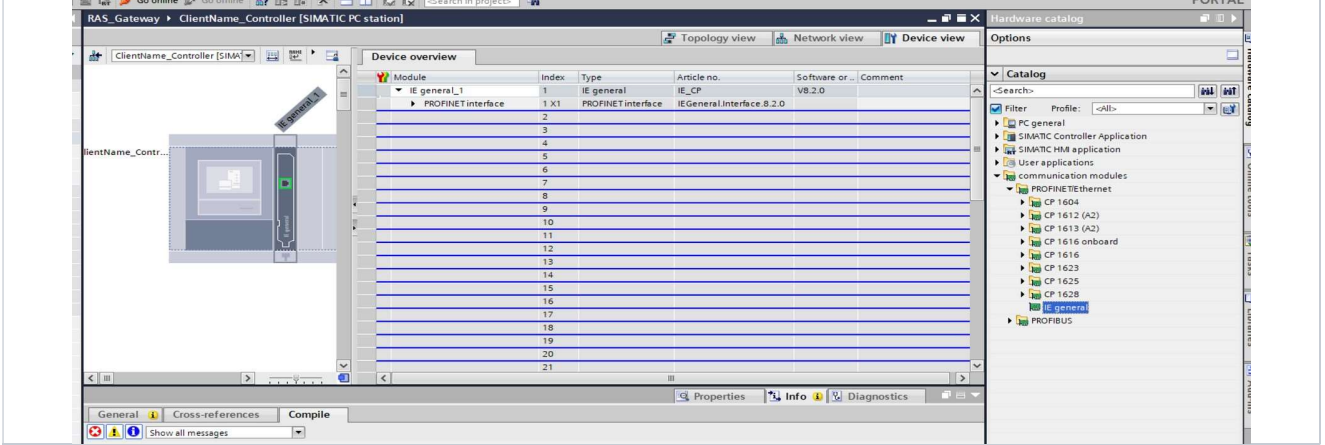
Project tree'de Add new device'ı açın, PC systems → PC general → PC station'ı seçin ve açıklayıcı bir ad verin (ör. <Müşteri/Tesis>_Controller).



Add new device → PC systems → PC general → PC station

2. Bir PROFINET arayüzü ekleyin (IE general)

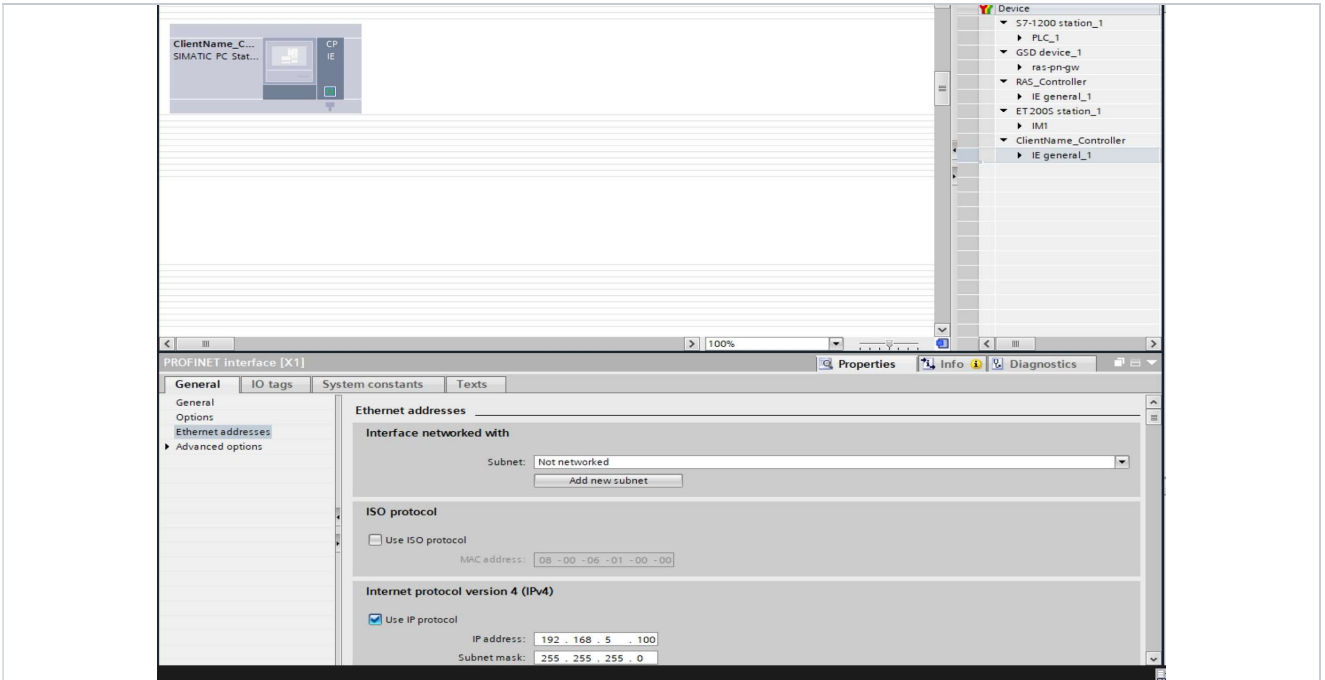
Donanım kataloğundan, Communication modules → PROFINET/Ethernet → IE general'ı PC istasyonunun rafına sürükleyin — bu modül, Controller'ın PROFINET tarafındaki ağ kartını temsil eder.



IE general modülü PC istasyonuna eklenmiş, donanım kataloğu sağda

3. Controller'ın IP adresini ayarlayın

Yeni arayüzün Properties → Ethernet addresses sayfasında, bu PC'nin gerçekte kullanacağı IP adresini ve alt ağ maskesini girin (gerekirse yeni bir alt ağ ekleyin) — IO-Device'lerle aynı alt ağda olmalıdır.

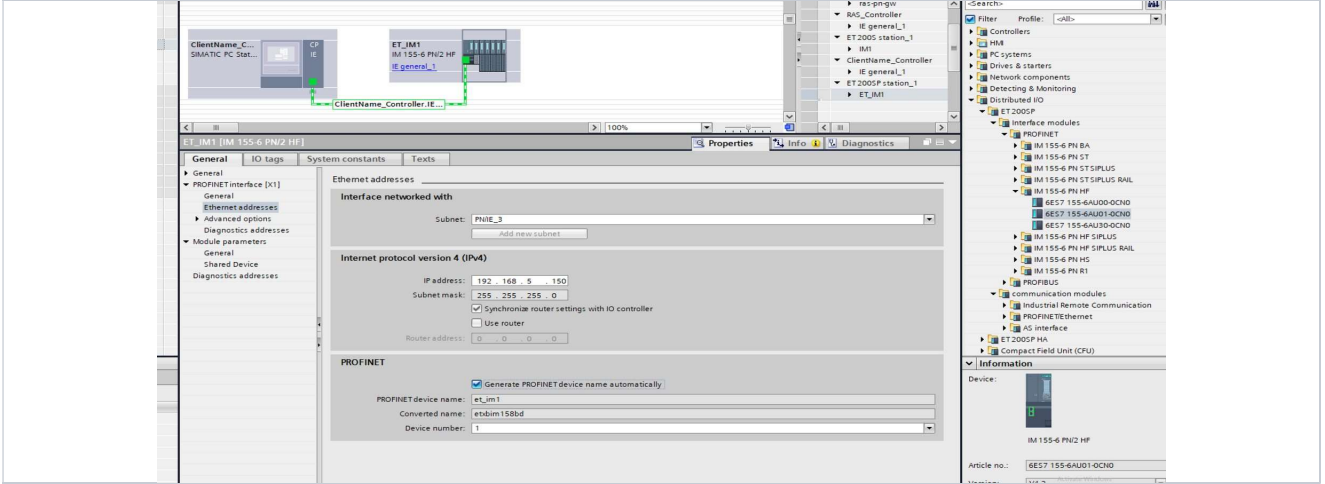


PROFINET arayüzü → Ethernet addresses: IP ve alt ağ

Not: Buraya girilen IP, TIA projesinin kendi iç referansıdır — TIA bunu gerçek bir ağ kartına atayamaz. Bu IP'nin Controller PC'sinin ağ kartına GERÇEKTEN atanmış olması (ve Controller'ın kendi ayarlarında, §9.6, aynı IP'nin görünmesi) sizin sorumluluğunuzdadır.

4. IO-Device'i ekleyip ağda bağlayın

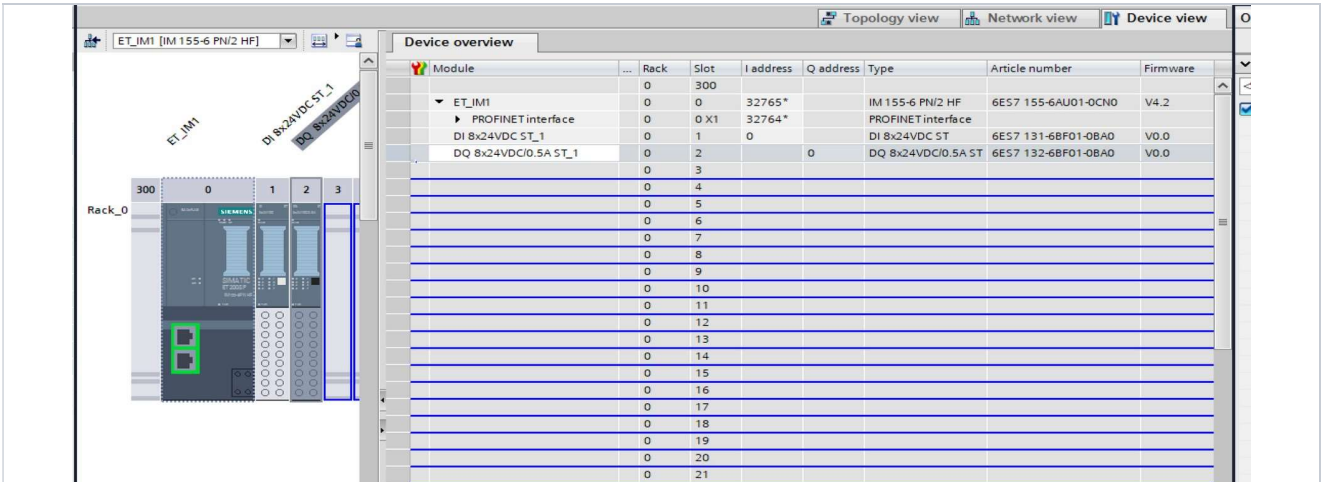
IO-Device'i projeye ekleyin (donanım kataloğunda henüz yoksa önce GSDML'sini içe aktarın), ardından Network view'da fare ile PC istasyonunun arayüzünden IO-Device'in arayüzüne bir PROFINET bağlantısı çizin — bu ikisini aynı IO sistemine sokar. IO-Device'in kendi istasyon adını ve IP'sini (Controller ile aynı alt ağ) kendi Ethernet addresses sayfasında ayarlayın.



Network view: PC istasyonu ↔ IO-Device bağlantısı; IO-Device'in Ethernet addresses sayfası

5. IO-Device'in modüllerini yapılandırın

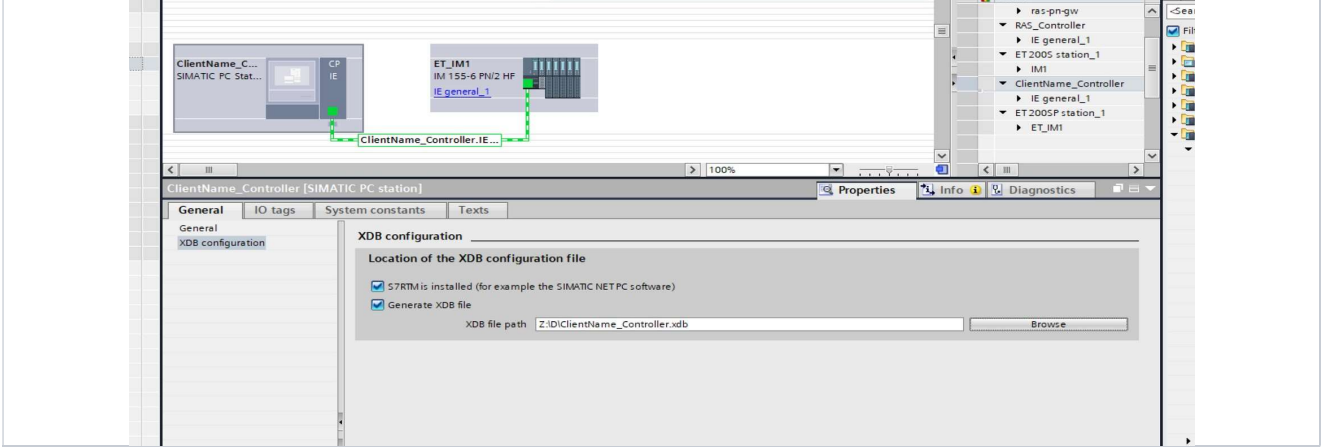
IO-Device'in kendi cihaz görünümünde, fiziksel olarak takılı modülleri (DI, DQ, analog vb.) takıldıkları gibi ekleyin — bu, .xdb dosyasına yansıyacak slot/alt slot düzenini ve giriş/çıkış uzunluklarını belirler.



IO-Device cihaz görünümü: takılı modüller (ör. DI 8x24VDC, DQ 8x24VDC/0.5A)

6. PC istasyonunda XDB üretimini etkinleştirin

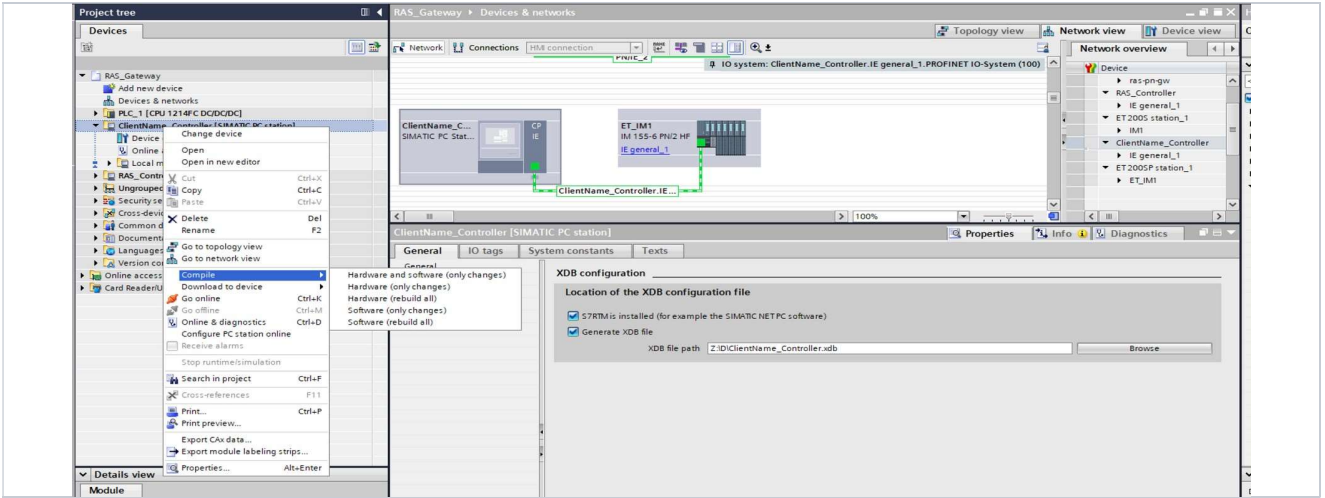
PC istasyonunda (IO-Device'te değil) Properties → General → XDB configuration'ı açın, Generate XDB file'ı işaretleyin ve yazılacağı yolu ayarlayın — Controller servisini çalıştıran PC'den erişilebilir bir konum.



PC istasyonu → Properties → XDB configuration: “Generate XDB file” ve dosya yolu

7. Derleyip .xdb dosyasını üretin

PC istasyonuna sağ tıklayıp Compile → Hardware (rebuild all)'ı seçin (sonraki küçük değişikliklerde Hardware (only changes) yeterlidir). TIA, derleme sırasında .xdb dosyasını yapılandırılan yola yazar.



PC istasyonuna sağ tık → Compile → Hardware (rebuild all)

İpucu: Aynı PC istasyonuna birden fazla IO-Device bağlıysa, TEK bir .xdb dosyası hepsini birden içerir — Controller'a aktarırken tümü tek seferde eklenir.

8. .xdb dosyasını Controller'a aktarın

Üretilen dosyayı Controller'ın web arayüzü yükleme alanına bırakın (§9.3). İçerik aktarma yanıtı, kaç cihazın içerik aktarıldığını ve varsa uyarıları listeler — ör. içerik aktarıcının tam yorumlayamadığı bir slot.

9.6 Controller ayarları ve ağ kartı

CONTROLLER AYARLARI

Cihazlara bağlanırken kullanılan istasyon adı ve IP. PN-IO RPC gerçek UDP/IP üzerinden gittiği için bu IP'nin ağ kartında gerçekten var olması gerekir.

Vendor ID	Device ID	PROFINET Katmanı	RPC Uç Noktası
0x0C00	0x0002	Çalışıyor	192.168.0.1:34964

İstasyon Adı	IP Adresi	Alt Ağ Maskesi
ras-controller	192.168.0.1	255.255.255.0

Ağ Geçidi

0.0.0.0

☒ IP'yi Windows ağ kartına da uygula (netsh ... add address, ek adres olarak)

Uygula

Controller ayarları — istasyon adı, IP/alt ağ/ağ geçidi ve IP'yi Windows ağ kartına da uygulama seçeneği

Bu, Controller'ın PROFINET ağındaki kendi kimliğidir: istasyon adı ve IP/alt ağ/ağ geçidi. Bu cihazın IP'si değişmez — Controller'ın DCP ve RPC yürütmek için kullandığı adrestir. RPC hatta gerçek bir adres gerektirdiğinden, "IP'yi Windows ağ kartına da uygula"ı işaretleyin ki adres seçili kartta gerçekten var olsun (ikincil adres olarak eklenir; mevcut adresler silinmez).

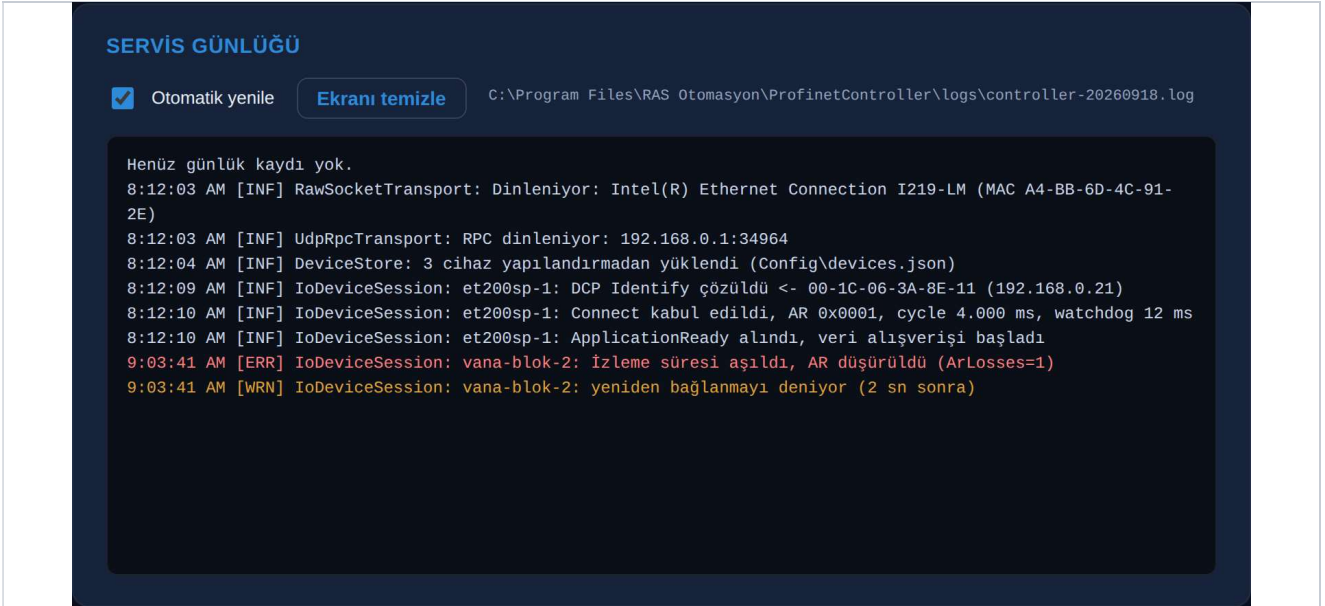
Not: Buradaki IP'yi değiştirmek PROFINET katmanını yeniden başlatır — cihazlar otomatik yeniden bağlanır ama kısa bir süre kopar. Bunu üretim hattı çalışırken değil, planlı bir pencerede yapın.

PROFINET katmanını elle durdurma ve başlatma

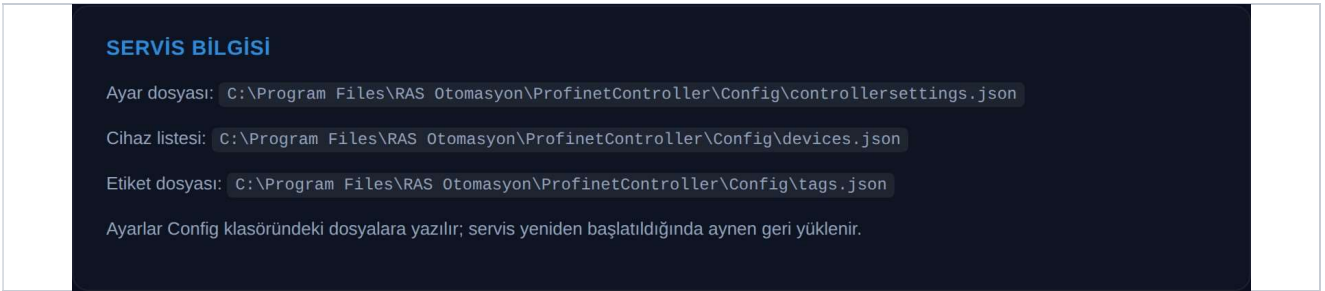
Aynı kart, tek bir cihazın Etkin kutucuğundan (§9.3) bağımsız çalışan tek bir Durdur/Başlat geçiş düğmesi taşır: Durdur, çalışmakta olan tüm cihaz oturumlarını kapatıp ardından ağ kartını ve RPC soketini tamamen kapatır (POST /api/stop); Başlat ise bunları yeniden açıp etkin olan her cihazı yeniden bağlar (POST /api/start). Durdur'u onaylamak tüm IO-Device'lerle bağlantıyı hemen keser ve Başlat'a tekrar basılana kadar hiçbir otomatik yeniden bağlanma denenmez — katman, bir operatör onu bilerek yeniden başlatana kadar kapalı kalır. PROFINET segmentini yeniden kablolamadan ya da gücünü kesip açmadan önce bunu kullanın, böylece Controller o süre boyunca başarılı olamayacak bağlantıları yeniden denemekle uğraşmaz.

Uyarı: Durdur yumuşak bir duraklama değil, sert bir kesmedir: IO-Device'lerin çevriminde güvenli bir noktayı beklemez. PROFINET kablosunu çekmekle aynı muamele edin — basmadan önce bağlı ekipmanın kendi mantığının çıkışların aniden kaybolmasına dayanabilecek bir durumda olduğundan emin olun.

9.7 Servis günlüğü ve yolları



Servis günlüğü kartı — bağlantı denemeleri, DCP çözümleme, hatalar ve yeniden bağlanmaların canlı, kayan görünümü



Servis bilgisi kartı — diskteki ayar, cihaz listesi ve etiket dosyalarının tam yolları

Otomatik yenile günlüğü iki saniyede bir sorgular; okurken görünümü dondurmak için işareti kaldırın (bu yalnızca ekrandaki listeyi temizler, diskteki günlük dosyasını asla). Servis bilgisi kartı, günlük dosyasını bir destek talebine eklerken ya da devices.json'u elle düzenlerken kullanışlıdır.

9.8 .NET istemci örneği

```
using RasProLink.Client;

using var client = new RasProLinkClient("localhost", 8080);
var status = await client.ConnectAsync();
Console.WriteLine($"{status.StationName}: {status.DeviceCountRunning}/{status.DeviceCount} devices running");

var detail = await client.GetDeviceAsync("et200sp-1");
var io = await client.GetDeviceIoAsync("et200sp-1");

await client.WriteDeviceOutputAsync("vana-blok-2", slot: 1, subslot: 1,
    data: new byte[] { 0x01 });

client.DevicesChanged += (s, list) => Console.WriteLine($"{list.Count} devices");
await client.StartPollingAsync();
```

9.9 Sorun giderme

Belirti	Kontrol
Web arayüzü hiç yüklenmiyor	Servis çalışıyor mu (services.msc)? 8080 portunu zaten başka bir şey mi kullanıyor?
Hiç ağ kartı listelenmiyor	Npcap kurulu değil, ya da servis başladıktan sonra kuruldu — kurup servisi yeniden başlatın.
Cihaz Resolving'den hiç çıkmıyor	DCP Identify yanıtı yok — kablolamayı ve seçili adaptörün (§4.3) ona gerçekten kablolandığını kontrol edin.
Cihaz Çalışıyor → Hata → Çalışıyor arasında dönüşüyor	İzleme süresine yaklaşıyor — cihaz ayrıntısı kartındaki (§9.4) jitter'ı yapılandırılmış güncelleme süresine karşı kontrol edin; genellikle meşgul/aşırı yüklü bir ağ segmenti.
XDB içe aktarma reddedildi	Bir cihaz vekil dışı aktarımı değil, ya da uyumsuz bir TIA Portal sürümünden dışı aktarılmış — yeniden dışı aktarıp tekrar deneyin.

9.10 GSDML kataloğu — TIA Portal ya da .xdb olmadan cihaz ekleme

Keşfedilen ya da katalogdan eklenen her cihaz da tıpkı §9.2'deki gibi aynı DCP/RPC/cyclic alışverişten geçer. GSDML kataloğu yalnızca bir cihazın slot/alt slot düzeninin Controller'a nasıl girdiğini değiştirir: §9.3 ve §9.5'te anlatılan TIA Portal + .xdb gidiş-dönüşünün yerine, üreticinin kendi GSDML dosyalarından oluşan bir klasör geçer.

Ağ Taraması (DCP)

Ağ Taraması bir DCP Identify-All broadcast'i gönderir ve yanıt veren her PROFINET istasyonunu listeler: 3. parti IO-Device'ler, başka Controller'lar, hatta Controller'ın kendi arayüzü. Bunların hiçbirisi için XDB ya da GSDML gerekmez — bu, TIA Portal'ın Erişilebilir cihazlar işlevine benzer şekilde çalışır. Her satır cihazın adını, IP'sini, MAC'ini ve Üretici/Cihaz Kimliğini gösterir; GSDML Eşleşmesi sütunu, o Üretici/Cihaz Kimliği için kataloğun en iyi eşleşmesini giriş/çıkış bayt toplamlarıyla birlikte gösterir, ya da hiçbirisi uymuyorsa Katalogda eşleşme yok yazar. Zaten yapılandırmada olan bir cihaz, Ekle yerine Yeniden yapılandır gösterir.

GSDML katalog klasörü

Katalog, arka planda bir klasördeki .xml/.gsdml dosyalarını tarar. Bu klasör varsayılan olarak, Servis bilgisi kartında (§9.7) görünen diğer dosyaların yanındaki Config\profinet-controller-gsdml klasörüdür ve Controller ilk başladığında otomatik oluşturulur. Üreticinin GSDML dosyasını/dosyalarını buraya bırakıp Katalogu Yeniden Yükle'ye tıklayın (ya da bir sonraki otomatik yeniden taramayı bekleyin) — servisi yeniden başlatmaya gerek yoktur. GSDML Katalogu kartı dosya sayısını, kaç cihaz tanımının ayrıştırıldığını, en son ne zaman yüklendiğini ve varsa dosya bazlı ayrıştırma hatalarını raporlar. Aynı karttan farklı bir klasör ayarlanabilir (yolu yazın, ya da Gözet'i kullanın — klasör seçici yalnızca servis etkileşimli bir masaüstü oturumunda çalışıyorsa işe yarar; yolu elle yazmak her zaman çalışır) ve bu, yeniden başlatmalar arasında hatırlanır.

Doğrudan katalogdan cihaz ekleme

Bir GSDML eşleşmesinin yanındaki Ekle'ye tıklamak — ister Ağ Taraması sonuçlarından ister katalogun kendi cihaz listesinden — o cihaz türüyle önceden doldurulmuş bir form açar: istasyon adını, IP/alt ağ/ağ geçidini girin ve katalogun isteğe bağlı takılabilir modüllerini slotlara atayın (Slot 0 her zaman sabit DAP'ın kendisidir).

Gönderildiğinde eksiksiz bir cihaz yapılandırması anında kurulur; normalde bir .xdb'nin taşıyacağı bağlantı parametreleri (AR, IOCR'lar, Alarm CR) Controller'ın elle kurulan her cihaz için kullandığı aynı varsayılanlarla doldurulur, böylece yeni cihaz tıpkı XDB'den içe aktarılmış gibi bağlanır. Takılabilir modülü olmayan kompakt bir cihaz slot tablosunu tamamen atlar.

Not: GSDML kataloğu yalnızca modül/slot düzenini sağlar; .xdb ile zaten eklenmiş bir cihazın üzerine asla yazmaz. İki yol serbestçe karıştırılabilir: bazı cihazlar bir .xdb'den içe aktarılmış, bazıları doğrudan katalogdan eklenmiş olarak aynı IO-Device'ler listesinde (§9.3) yan yana durabilir.

Bölüm 10

PROFINET Device

10.1 Genel bakış

PROFINET Device protokol katmanı, Controller'ın ayna görüntüsüdür: PC'nizi gerçek bir PLC için bir PROFINET IO-Device'ına — bir Slave'e — dönüştürür. RasProLink, Device lisansınızla birlikte gelen GSDML dosyasındaki modül/slot düzenine eşleşen bir cihazı simüle eder; slot yapılandırmanız bu GSDML'yi yansıttığında, PLC IO haberleşmesi PLC tarafında programlama gerektirmeden, yalnızca normal bir donanım yapılandırması içe aktarımıyla çoktan hazır hale gelir.

Öge	Değer
Rol	IO-Device (Slave)
Tarafından tanımlanır	Device lisansı ile birlikte gelen bir GSDML dosyası (RasProLink dışında bakımı yapılır, RasProLink tarafından üretilmez)
PLC tarafından keşif	DCP (istasyon adı, gerekirse IP)

10.2 GSDML — nedir ve nasıl kullanılır

GSDML (General Station Description Markup Language), her PROFINET cihazının kendini tanımlamak için yayınladığı XML lehçesidir; kimliğini (Vendor ID, Device ID), modül kataloğunu ve her modülün giriş/çıkış veri düzenini içerir. Bir PLC mühendisi bu tek dosyayı TIA Portal'a (ya da başka herhangi bir PROFINET mühendislik aracına) içe aktarır ve cihaz, donanım kataloğunda tam olarak gerçek bir Siemens/üretici parçası gibi görünür — ağa sürüklenebilir, modülleri yapılandırmaya hazır.

RasProLink bir GSDML dosyası üretmez ya da dışa aktarmaz — bu dosya, ürünün dışında bakımı yapılan ve RasProLink Device lisansınızla birlikte gelen sabit bir dosyadır. RasProLink'in yaptığı şey, o dosyanın modül listesini birebir yansıtmaya gereken bir slot yapılandırması kurmanıza izin vermektir: aşağıdaki modül kataloğu RasProLink tarafında sabittir ve yalnızca TIA Portal'ın (ya da başka bir mühendislik aracının) içe aktardığı GSDML'de aynı modüllerin tanımlandığı durumda kullanılabilir. Gerçekten yeni bir modül boyutu eklemek bir yapılandırma ekranı işlemi değildir — önce GSDML dosyasının kendisinin güncellenip TIA Portal'a yeniden içe aktarılması gerekir; ancak bundan sonra eşleşen boyut burada seçilebilir hale gelir.

Modül	Giriş / Çıkış
8 Byte In/Out	8 bayt giriş, 8 bayt çıkış
16 Byte In/Out	16 bayt giriş, 16 bayt çıkış
32 Byte In/Out	32 bayt giriş, 32 bayt çıkış
64 Byte In/Out	64 bayt giriş, 64 bayt çıkış

Bu dört simetrik modül kataloğun tamamıdır — slot 0 her zaman sabit DAP'ın kendisidir, azami 8 veri slotu (1–8 slotları) bu dört boyuttan birini alabilir. Bu, çoğu nokta-noktaya veri alışverişi ihtiyacını kutudan çıkar çıkmaz karşılar; her slot için yükü karşılayan boyut seçilir. Uygulamanız asimetrik ya da daha büyük bloklara ihtiyaç

duyuyorsa, bu bir GSDML değişikliğidir ve — yukarıda anlatıldığı gibi — yalnızca RasProLink üzerinden yapılandırılmaz, GSDML düzeyinde talep edilip devreye alınması gerekir.

10.3 Gerçek bir PLC'ye karşı devreye alma

6. RasProLink Device lisansınıza ait GSDML dosyasını edinin (bkz. §10.2) ve donanım kataloğunda henüz yoksa TIA Portal'a içe aktarın: Options → Manage general station description files (GSD).
7. RasProLink'in web arayüzünde, Device'in istasyon adını ayarlayın ve her slot için, içe aktarılan GSDML'deki karşılık gelen slotla eşleşen modül boyutunu seçin — bkz. §10.2'deki modül kataloğu.
8. PLC projesinde, cihazı donanım kataloğundan ağa sürükleyin, PLC'nin PROFINET arayüzüne bağlayın; aynı istasyon adını verin ve PLC'nin alt ağında bir IP atayın.
9. Donanım yapılandırmasını PLC'ye indirin. PLC, Device'ı DCP üzerinden istasyon adıyla çözer, bağlanır ve cyclic veri akmaya başlar — bir IO-Device'in Controller'da görünmesiyle aynı şekilde (§9.2), RasProLink'in kendi cihaz durumunda da görünür.

10.4 .NET istemci örneği

```
using RasProLink.Client;

using var client = new RasProLinkClient("localhost", 8080);
await client.ConnectAsync();

// Read what the PLC most recently wrote to us (its outputs, our inputs)
var io = await client.GetDeviceIoAsync("self");

// Publish new input data for the PLC to read next cycle
await client.WriteDeviceOutputAsync("self", slot: 1, subslot: 1,
    data: new byte[] { 0x00, 0x2A });
```

10.5 Sorun giderme

Belirti	Olası neden
PLC cihazı hiç bulamıyor	İstasyon adı uyuşmuyor (büyük/küçük harf duyarlı, yalnızca DNS-uyumlu karakterler), ya da PLC ve Device aynı alt ağda değil.
PLC'de modül uyumsuzluğu alarmı	RasProLink'te yapılandırılan slot düzeni artık PLC projesine içe aktarılan GSDML ile eşleşmiyor — her slotu GSDML dosyasının modül listesine (§10.2) göre yeniden kontrol edip uyuşmayan slot(lar)ı düzeltin.

10.6 Tanılama

Tanılama kartı bir devreye alma ve kablolama doğrulama aracıdır: Device'ın PLC'ye gönderdiği veriye bilinen test desenleri enjekte etmenizi, tek bir etiketi ya da baytı sabit bir değere kilitlemenizi, bir modül arızasını simüle etmenizi ve çalışan çevrim istatistiklerini sıfırlamanızı sağlar — hepsi PLC programına dokunmadan. Bu araçlar PLC'nin gerçekte aldığı veriyi etkiler, bu yüzden bunlar tezgah/devreye alma yardımcıları olarak ele alınmalıdır:

bir force ya da test deseni açırken, kendi uygulamanızın yazdığı değerler ezilir ve sayfayı kapattığınızda RasProLink bunları otomatik olarak kapatmaz.

Test desenleri

PLC tarafından verinin uçtan uca gerçekten aktığını doğrulamak için bir desen, bir slot (ya da ilk uygun slotta bırakın) ve bir güncelleme aralığı seçip uygulayın: Sayaç, slotun giriş verisinin ilk baytını her aralıkta bir artırır; Rampa, bayt $n = \text{sayaç} + n$ yazar, bu da PLC tarafındaki eşleme yanı sıra bir ofset kaymasını da ortaya çıkarır; Değişken, tüm alt modülü 0x00 ile 0xFF arasında değiştirir; Rastgele, rastgele baytlar yazar ve yalnızca veri yolunun şeffaf olduğunu (arada bir baytın takılıp kalmadığını ya da maskelenmediğini) doğrulamak için kullanışlıdır. Aynı anda yalnızca bir desen çalışır; Kapalı'ya geçmek normal uygulama verisini geri getirir.

Sinyal force

Force etmek, seçilen bir giriş etiketini, girdiğiniz bir değere (hex olarak) sabitler — test deseninden ve altta kendi uygulamanızın ne yazdığından bağımsız olarak. Altındaki değer silinmez — yazılmaya ve izlenmeye devam eder, yalnızca hat üzerinde geçersiz kılınır — bu yüzden force'u kaldırmak (etiket bazlı ya da Tümünü Kaldır) gerçek değerini uzlaştıracak bir şey olmadan hemen geri görünmesini sağlar. Etkin force'lar tablosu her birinin slotunu, ofsetini, etiket adını ve force edilen değerini bir Kaldır eylemiyle listeler; bu, gerçek saha kabloları bağlanmadan önce belirli bir alt akış sinyalinin (bir alarm biti, bir analog okuma) beklenen PLC davranışını tetiklediğini doğrulamak için kullanılan araçtır.

Simüle edilmiş arıza ve istatistik sıfırlama

"Modül arızası simüle et (IOPS = Bad)", hiçbir şeyin bağlantısını kesmeden alt modülün IOPS durum baytını Bad'e zorlar — bu, PLC programının, proses görüntüsünde ne bayt olursa olsun sessizce güvenmek yerine kötü bir veri kalitesi göstergesine (ör. bir modül arızası OB'si, alt akış mantığını kapatan bir kalite biti) gerçekten tepki verdiğini kontrol etme yöntemidir. Normal (Good) durumu geri getirmek için tekrar kapatın. İstatistikleri sıfırla, arayüzün başka yerlerinde gösterilen çevrim sayaçlarını (gönderilen/alınan çerçeveler, jitter vb.) sıfırlar; bu, önceki başlangıç geçişlerinin sayıları çarpıtmaması için zamanlanmış bir test koşusundan hemen önce kullanışlıdır.

Not: Kartın üstündeki tanımlama istatistik kutuları (Force Edilen Bayt, Test Deseni), o an neyin geçersiz kılındığının bir bakışta özetini verir — bir istasyonu teslim etmeden önce bunları kontrol edin, devreye alma sırasında force edilen hiçbir şey üretimde etkin kalmasın. Altta yatan REST uç noktaları (GET/POST /api/diagnostics ve alt rotaları) Ek A'da listelenmiştir.

Bölüm 11

.NET İstemci Kütüphanesi

RasProLink.Host, tek bir süreçte beş protokol katmanından (EtherCAT Master, EtherNet/IP Scanner, EtherNet/IP Adapter, PROFINET Controller, PROFINET Device) herhangi birini barındırabilir, ama bir anda yalnızca biri aktiftir — web arayüzündeki protokol seçim ekranından protokol katmanı değiştirildiğinde süreç o protokol katmanı ile yeniden başlar (§4.4). Aktif protokol katmanının REST yüzeyi katmandan katmana farklıdır; bu fark en belirgin biçimde, bir etiket yazma isteğinin hangi hedefe (cihaz/slave/scanner hedefi) gittiğini nasıl belirttiğinde ortaya çıkar:

Protokol katmanı	Etiket yazma ucu
EtherCAT Master	POST /api/ethercat/slaves/{id}/tags/write — {id}, slave'in iç GUID'idir.
EtherNet/IP Scanner	POST /api/ethernetip/scanner/targets/{id}/tags/write — {id}, hedefin iç GUID'idir.
PROFINET Controller	POST /api/devices/{name}/tags/write — {name}, PROFINET istasyon adıdır, GUID yoktur.
EtherNet/IP Adapter	POST /api/ethernetip/adapter/tags/write — hedef/id yok, tek sabit yerel IO kümesi.
PROFINET Device	POST /api/io/write-tags — hedef/id yok, tek sabit yerel IO kümesi.

RasProLink.Client bu farkı tamamen gizler. WriteTagAsync dahil tüm ortak metotlar, hangi protokol katmanı aktif olursa olsun birebir aynı imzada kalır: iç tarafta ConnectAsync (GET /api/engine) ile hangi katmanın aktif olduğu öğrenilir ve her çağrı o katmanın gerçek REST yoluna çevrilir (EtherCAT/Scanner için ad→GUID eşlemesi GetTargetsAsync ile bir kez çıkarılıp önbelleklenir). Şu satır

```
await client.WriteTagAsync("IM1", "Motor_Run", true);
```

RasProLink.Host'ta hangi protokol katmanı çalışırsa çalışsın değişmeden çalışmaya devam eder (EtherNet/IP Adapter ve PROFINET Device'ta "IM1" hedef adı yok sayılır, çünkü bu iki katmanın tek sabit bir yerel IO kümesi vardır). Bu ortak akışa uymayan katmana özgü bir işlev (ör. ham EtherCAT SDO erişimi, PROFINET .xdb içe aktarma) bunun yerine ayrı, katmana özgü bir metot olarak eklenir — WriteTagAsync/ReadTagsAsync'in kendi davranışını hiç değiştirmez.

Bu tasarımın ötesinde kütüphane, kendi .NET uygulamanızın web arayüzünü sürmek yerine servise bağlanmak, protokol katmanı durumunu ve hedefleri okumak, kodda etiket okuyup yazmak için referans aldığı bağımlılıksız, küçük bir pakettir. İki DLL olarak gelir:

Derleme	İçerik
RasProLink.Client.Abstractions.dll	Paylaşılan, protokol katmanından bağımsız veri tipleri (EngineKind, EngineStatus, TargetSummary, TagDefinition, TagValue...) ve kendi içinde bağımsız küçük bir JSON okuyucu/yazıcı.
RasProLink.Client.dll	Gerçek istemci (RasProLinkClient), HttpClient üzerine kurulu.

Not: İki DLL'nin de hiçbir NuGet bağımlılığı yoktur ve ikisi de hem netstandard2.0 hem net8.0'ı hedefler — aynı kütüphane, eski bir .NET Framework 4.7.2 WinForms uygulamasından modern bir .NET 8 servisine kadar, binding-redirect sürprizi olmadan değişmeden çalışır.

11.1 Kurulum ve hızlı başlangıç

Projenizden her iki DLL'i de referans alın (bir NuGet akışı gerekmez — düz dosya referansı yeterlidir ya da projeleri kaynaktan derleyip proje referansı ekleyebilirsiniz).

```
using RasProLink.Client;
using RasProLink.Client.Abstractions;

using var client = new RasProLinkClient("localhost", 8080);

var active = await client.ConnectAsync(); // GET /api/engine – which engine is
active
Console.WriteLine($"Active engine: {active}");

var status = await client.RefreshStatusAsync();
Console.WriteLine($"{{status.RoleDisplayName}}: enabled={{status.Enabled}}
connected={{status.Connected}}");

var targets = await client.GetTargetsAsync();
foreach (var t in targets)
    Console.WriteLine($"{{t.Name}} @ {{t.IpAddress}} [{{t.State}}] in={{t.InputBytes}}B
out={{t.OutputBytes}}B");
```

ConnectAsync'in açıkça çağırılması gerekmez — diğer her metot, gerekirse ilk kullanımda bunu kendiliğinden yapar. Kendiniz çağırarak yalnızca aktif protokol katmanını önceden bilmek/loglamak istediğinizde işe yarar; ondan sonra IsConnected ve ActiveEngine, yeni bir istek göndermeden önbellekteki sonucu okur.

11.2 Protokol katmanı durumu ve hedefler

```
var info = await client.GetEngineInfoAsync(); // full engine-selection list, not just the
active one
Console.WriteLine($"Active: {info.Active}");
foreach (var opt in info.Available)
    Console.WriteLine($" {{opt.DisplayName}} ({{opt.Kind}}) productionReady={{opt.ProductionReady}}");

var status = await client.RefreshStatusAsync(); // normalized snapshot of the ACTIVE
engine
Console.WriteLine(status.Message ?? "(no message)");
```

GetEngineInfoAsync, web arayüzünün protokol seçim ekranının (§4.4) gösterdiği aynı listeyi döner — RasProLink.Host'un geçebileceği her protokol katmanı, yalnızca aktif olan değil; kendi motor seçiminizi kurmak ya da bir geçiş sunmadan önce ProductionReady'yi kontrol etmek için kullanılabilir. Buna karşılık RefreshStatusAsync her zaman yalnızca o an aktif olan protokol katmanını anlatır, ortak EngineStatus şekline indirgenmiş olarak: Enabled, katmanın bilerek durdurulup durdurulmadığını yansıtır (bkz. §11.8); Connected katmana göre farklı bir anlama gelir (EtherCAT/PROFINET'te veriyolu/PROFINET katmanı ayakta, Adapter'da bir scanner bağlantısının var olması) — bu, herhangi bir hedefin değil, katmanın TÜMÜNÜN özetidir. Hedef bazında durum bunun yerine TargetSummary.State'te bulunur.

GetTargetsAsync, protokol katmanının cihaz/slave/scanner hedefi listesini TargetSummary'ye indirgenmiş olarak döner — Name, aşağıdaki etiket metotlarına target argümanı olarak geçirilecek değerdir. EtherNet/IP Adapter ve PROFINET Device'ta gerçek bir hedef listesi yoktur (tek sabit yerel IO kümesi), bu yüzden GetTargetsAsync bu ikisinde onun yerine tek, sentetik bir satır döner.

11.3 Etiket yazma

```
// "IM1" is a target Name from GetTargetsAsync (§11.2). On EtherNet/IP Adapter
// and PROFINET Device it is accepted but ignored — there is only one target.
await client.WriteTagAsync("IM1", "Motor_Run", true);

// Write several output tags on one target in a single round trip
await client.WriteTagsAsync("IM1", new Dictionary<string, object?>
{
    ["Motor_Run"] = true,
    ["Speed_Setpoint"] = 1500,
});
```

WriteTagAsync tek bir etikete yazar; WriteTagsAsync üzerine kurulu, tek girdili ince bir sarmalayıcıdır. Aynı hedefte birden fazla etiket ayarlarken WriteTagsAsync'i tercih edin — birkaç istek yerine tek istek, hem daha az ağ trafiği hem de değerlerin farklı protokol çevrimlerine düşme riskinin olmaması demektir. Yalnızca Çıkış yönündeki etiketler yazılabilir — tam olarak web arayüzünün kendi kuralı gibi (§5.3) — bir Giriş etiketine ya da o hedefte var olmayan bir etikete yazmak RasProLinkClientException fırlatır (§11.6).

11.4 Canlı güncelleme — polling

```
client.StatusChanged += (s, status) => Console.WriteLine($"status: {status}");
client.TargetsChanged += (s, list) => Console.WriteLine($"{{list.Count}} targets");
client.PollError += (s, ex) => Console.WriteLine($"poll error: {ex.Message}");

client.PollIntervalMs = 500; // default 1000
await client.StartPollingAsync();
// ...
client.StopPolling();
```

Servis yalnızca REST sunar, push kanalı yoktur, bu yüzden istemci kütüphanesi polling yapar: StartPollingAsync, her PollIntervalMs'te RefreshStatusAsync ve GetTargetsAsync'i çağıran bir arka plan döngüsü çalıştırır ve her turdan sonra StatusChanged ve TargetsChanged'i tetikler — web arayüzünün kendisinin de sorguladığı aynı bilgidir. Başarısız bir tur (servise kısa süreli ulaşılabilmesi) döngüyü durdurmak yerine PollError'ı tetikler; polling bir sonraki tik'te yeniden denemeye devam eder. Etiket değerleri polling'in parçası değildir — hangi etiketlerin önemli olduğu ve ne sıklıkla okunması gerektiği uygulamaya özgü olduğundan, ReadTagsAsync'in kendi zamanlamasına göre çağırılması gerekir.

11.5 Koddan etiketler

```
var defs = await client.GetTagDefinitionsAsync("IM1");

await client.AddOrUpdateTagAsync("IM1", new TagDefinition
{
    Name = "Sensor_Input1",
    Direction = IoDirection.Input,
    Type = TagDataType.Bool,
    Offset = 0,
    BitOffset = 0
});

await client.RemoveTagAsync("IM1", "Obsolete_Tag");
```

```
// One round trip fills a local table for GetTag/ReadTag<T> below
var values = await client.ReadTagsAsync("IM1");

bool running = client.ReadTag<bool>("IM1", "Motor_Run");
TagValue? raw = client.GetTag("IM1", "Sensor_Input1");
Console.WriteLine(raw?.Text);
```

GetTagDefinitionsAsync, AddOrUpdateTagAsync, SetTagDefinitionsAsync (bir hedefin tüm etiket tablosunu tek çağrıda değiştirir) ve RemoveTagAsync, web arayüzündeki etiket düzenleyiciyi yansıtır (§5.3) — Slot/Subslot yalnızca PROFINET protokol katmanlarında anlamlıdır, diğerlerinde varsayılanlarında (0/1) bırakılır.

ReadTagsAsync, değerler için ağa giden tek çağrıdır; hedef ve etiket adına göre anahtarlanmış yerel bir tabloyu doldurur. GetTag ve generic ReadTag<T> ise bu yerel tablodan okur — hiçbir zaman istek göndermezler, bu yüzden daha güncel bir değere ihtiyacınız olduğunda ReadTagsAsync'i tekrar çağırın. ReadTag<T>, TagValue'nun sakladığı sayı/metin/ham baytları bool, string, byte[] ya da sayısal bir tipe çevirir; Int64/UInt64 etiketler için bunun yerine doğrudan TagValue.Raw'ı okuyun, çünkü bir double, 64 bitlik aralığın tamamını kayıpsız tutamaz.

11.6 Hata yönetimi

Başarısız her çağrı bir RasProLinkClientException fırlatır. Servis yapılandırılmış bir hata gövdesiyle yanıt verdiğinde, mesajı gerçek nedendir; bir etiket metoduna geçirilen bir hedef adı çözümlenemediğinde (yalnızca EtherCAT Master / EtherNet/IP Scanner — bkz. §11.2), istemci bunu sunucuya hiç gitmeden doğrudan bildirir; servise hiç ulaşamadığında, mesaj bunu söyler ve alttaki taşıma istisnasını sarar.

11.7 Kütüphane olmadan — düz HTTP

Kütüphane ince bir sarmalayıcıdır — her işlem düz bir HTTP çağrısıdır, herhangi bir dilden ulaşılabilir. Tam uç nokta listesi Ek A'da; işte bir PROFINET Controller için aynı “cihazları listele” çağrısı curl ve Python'da — diğer dört protokol katmanı için alttaki yol, bu bölümün başında özetlendiği gibi farklıdır.

```
curl http://localhost:8080/api/devices
```

```
import requests
devices = requests.get("http://localhost:8080/api/devices").json()
for d in devices:
    print(d["name"], d["ipAddress"], d["state"])
```

RasProLink ile birlikte gelen examples klasöründe, beş dilde — C#, VB.NET, Python, PowerShell ve Node.js — tam bir 16 kanallı yürüyen bit testi istemcisi bulunur; her biri kendi istemci kütüphanesi kopyasıyla bağımsız olarak derlenip çalıştırılmaya hazırdır.

11.8 Protokol katmanını durdurma ve başlatma

```
await client.StopAsync(); // same effect as the web interface's Stop control – halts ALL targets
// ...
await client.StartAsync(); // resumes it
```

StopAsync ve StartAsync, web arayüzünde aktif protokol katmanının kendi durum sayfasında gösterilen Durdur/Başlat denetiminin istemci tarafındaki karşılığıdır — tek bir etiketi ya da hedefi değil, aktif katmanın

TÜMÜNÜ, tüm hedefleriyle birlikte durdurur ya da devam ettirir. StopAsync'ten sonra RefreshStatusAsync'in EngineStatus.Enabled'ı false olur; Connected da düşer, çünkü artık hiçbir cyclic veri alışverişi yapılmamaktadır. Tek bir hedefi durdurmak, devre dışı bırakmak ya da kaldırmak için bir istemci metodu yoktur — bu ayrıntı düzeyi bu kütüphane üzerinden değil, her protokol katmanının web arayüzündeki kendi cihaz/hedef düzenleyicisi üzerinden yapılandırılır.

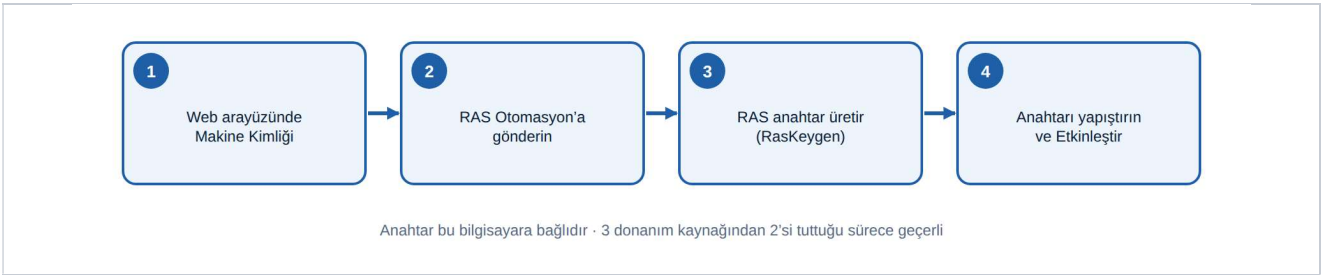
Bölüm 12

Lisanslama

12.1 Deneme süresi

RasProLink, beş protokol katmanının herhangi birinde, hiçbir lisans girilmeden ilk başlangıçtan itibaren 30 gün boyunca tam özellikli çalışır: her protokol katmanı, her cihaz için sınırsız. Deneme süresi bir anahtar girilmeden bittiğinde aktif protokol katmanı durur (cihazların bağlantısı kesilir), ama web arayüzü çalışmaya devam eder; böylece bir anahtar hâlâ yapıştırılabilir.

12.2 Lisans alma ve etkinleştirme



Anahtar bu bilgisayara bağlıdır; 3 donanım kaynağından 2'si tuttuğu sürece geçerlidir.

10. Lisans kartını (§4.2) açın ve Makine Kimliği'ni kopyalayın.
11. Sipariş/müşteri referansı ile birlikte RAS Otomasyon'a gönderin.
12. RAS Otomasyon, o Makine Kimliği için RasKeygen ile imzalı bir anahtar üretir (RasKeygen, RAS Otomasyon'dan hiç çıkmayan dahili bir araçtır).
13. Anahtarı Lisans kartına yapıştırıp Etkinleştir'e tıklayın.

Bir lisans, aynı Lisans kartından ya da DELETE /api/license ile de kaldırılabilir; bu işlem kayıtlı anahtarları siler ve servisi deneme durumuna döndürür (§12.1). Bu seçenek destek ve devretme durumları (ör. PC'yi başka birine devretmek) içindir, rutin etkinleştirmede gerekmez.

12.3 Lisans neye bağlıdır

Anahtar, belirli PC'nin birkaç bağımsız donanım/işletim sistemi tanımlayıcısı üzerinde bir parmak izidir. Bu kaynaklardan 3'ünün 2'si eşleşmeye devam ettiği sürece geçerliliğini korur; başka bir makineye toptan taşınmasına izin vermeden, rutin bir değişikliğe (yalnızca bir işletim sistemi yeniden kurulumu gibi) tolerans gösterir.

Bir lisans, o PC'deki her protokol katmanını etkinleştirir; protokol başına ayrı lisans yoktur. Makine Kimliği üzerinde bir kez etkinleştirme yapmak, hangi protokol katmanının aktif olarak seçildiğine bakılmaksızın EtherCAT Master, EtherNet/IP Scanner, EtherNet/IP Adapter, PROFINET Controller ve PROFINET Device'in hepsini kapsar.

12.4 Lisans hata mesajları

Mesaj	Anlamı
Anahtar bu makineye ait değil	Anahtar farklı bir Makine Kimliği için üretilmiş — talep ederken neyin gönderildiğini yeniden kontrol edin.
Lisans dosyası yazılamadı	Servis hesabı Config klasörüne yazamıyor — klasör izinlerini kontrol edin.
Boş anahtar	Etkinleştir'e tıklamadan önce alana hiçbir şey yapıştırılmadı.

Bölüm 13

Performans ve Gerçek Zamanlı Davranış

13.1 Neden önemli

Windows gerçek zamanlı bir işletim sistemi değildir: jenerik bir zamanlayıcı üzerine kurulan saf bir cyclic döngü, normal masaüstü zamanlaması altında amaçlanan periyottan birkaç milisaniye kayabilir; bu kayma, hızlı bir saha yolunda çevrim süresinin kendisinden bile büyük olabilir. RasProLink'in beş protokol katmanı da, sıradan PC donanımında bu kaymayı çevrim süresinin bir kısmına indirmek için aynı ortak teknik kümesini paylaşır; bunun için gerçek zamanlı bir işletim sistemi yaması ya da özel bir donanım zamanlayıcı kartı gerekmez.

13.2 Dört teknik

- Özel, yüksek öncelikli thread — cyclic alışveriş, sürecin yaptığı diğer her şeyle thread pool'u paylaşmak yerine kendi Highest öncelikli thread'inde çalışır.
- Windows multimedia-timer çözünürlüğü — sistem geneli zamanlayıcı tik'i, cyclic döngünün ömrü boyunca geçici olarak 1 ms'e yükseltilir (varsayılan ≈ 15.6 ms'den), böylece kaba bir bekleme gerçekten gerektiği zamana yakın uyanır.
- CPU çekirdeği sabitleme — cyclic thread son mantıksal CPU çekirdeğine sabitlenir, böylece işletim sistemi onu çevrim ortasında asla taşımaz; her taşıma önbellek/TLB durumunu sıfırlar ve tam da zamanlamanın en önemli olduğu anda yüzlerce mikrosaniyelik jitter ekleyebilir.
- Çevrim başına sıfır heap tahsisi — IO tamponları ve cihaz listeleri bir kez tahsis edilir ve her çevrimde yeniden kullanılır (yerinde temizlenip yeniden yazılır), çöp toplayıcı duraklamalarını zamanlama açısından kritik yoldan kaldırır.

13.3 Ne beklemeli

Bu dört tekniğin uygulandığı tipik bir Windows PC'de, EtherCAT Master jitter'ı 4 ms çevrimde azami ≈ 0.3 ms olarak ölçülmüştür; bu, ince ayardan önceki 9–10 ms'lik değere göre belirgin bir düşüştür. Sonuçlar belirli PC'ye (CPU nesli, çekirdek sayısı, güç planı) ve aynı anda çalışan diğer işlemlere bağlıdır, ama aynı dört teknik beş protokol katmanının hepsinde aynı şekilde uygulandığından PROFINET ve EtherNet/IP çevrimlerinde de benzer bir iyileşme beklenir.

13.4 Arayüzde jitter değerini okuma

Her cihazın istatistiklerinde gösterilen çevrim jitter'ı (PROFINET Controller için §9.4; aynı alan her protokol katmanının cihaz ayrıntısında mevcuttur), bir cihazın gerçek cyclic zamanlamasının yapılandırılmış güncelleme süresinden ne kadar saptığını ölçer. Jitter yapılandırılmış watchdog/zaman aşımına yaklaştıkça ara sıra bağlantı kopması olasılığı artar; çözüm genellikle o cihaza daha uzun bir güncelleme süresi tanımlamak (çevrim başına daha fazla pay) ya da saha tarafı ağ segmentindeki tıkanıklığı azaltmaktır (daha az cihaza hizmet veren hub benzeri paylaşılan bir anahtar yerine düzgün QoS'lu bir anahtara geçmek gibi).

Bölüm 14

Genel Sorun Giderme

Protokole özgü sorunlar her protokol katmanının kendi bölümünde ele alınır (§6.5, §7.4, §8.4, §9.9, §10.5).

Aşağıdakiler beş protokol katmanının hepsinde ortaktır.

Belirti	Kontrol
Web arayüzüne başka bir PC'den ulaşılamıyor	Windows Güvenlik Duvarı gelen TCP 8080'i engelliyor — izin verin (§3.3).
Protokol katmanını değiştirme takılı kalıyor gibi	Servis, değiştirmede dahili olarak yeniden başlar — birkaç saniye bekleyip yenileyin; yaklaşık 30 sn'den uzun sürerse services.msc'yi kontrol edin.
Temiz bir yeniden kurulumdan hemen sonra lisans deneme süresi bitti diyor	uninstall.bat -RemoveConfig ile çalıştırılmış, deneme sayacını diğer her şeyle birlikte silmiş — denemeyi tekrarlamak yerine bir lisans için RAS Otomasyon'la iletişime geçin.
Bir destek talebine tanılama eklemek gerekiyor	Servis bilgisi kartının yollarını (§9.7) kullanarak güncel günlük dosyasını ve devices.json'u ekleyin.

Ek A

HTTP API Referansı

Aşağıdaki her uç nokta, protokol katmanının taban URL'si altında (varsayılan olarak `http://<pc>:8080`) düz bir JSON REST çağrısıdır; alan adları camelCase'dir ve yerleşik bir kimlik doğrulama yoktur — servis güvenilir bir yerel ağda çalışmak üzere tasarlanmıştır. Güvenilmeyen bir ağdan ulaşılması gerekiyorsa, kimlik doğrulama ekleyen bir ters proxy'nin arkasına koyun. Uç nokta kümesi yeniden başlatmalar ve protokol katmanı değişimleri arasında aynıdır; yalnızca Cihazlar bölümü her protokol katmanına göre, o protokolün kendi cihaz modelini yansıtacak şekilde farklıdır.

A.1 Paylaşılan uç noktalar — beş protokol katmanının tümü

Uç nokta	Amaç
GET /api/status	Protokol katmanı kimliği, aktif protokol ve özet sayaçlar.
GET /api/host/info	Ürün adı, sürüm, hangi protokol katmanının aktif ve üretime hazır olduğu, servisin başlama zamanı — hangi protokol katmanının çalıştığından bağımsız (§4).
GET /api/engine POST /api/engine	Beş protokol katmanını üretime hazır olma durumlarıyla listele, ya da aktif olanı değiştir; değişiklik enginesettings.json'a kaydedilir ve servis birkaç saniye içinde yeni protokol katmanıyla yeniden başlar (§4).
GET/POST /api/settings/language	Arayüz/günlük dilini (tr/en/de) oku ya da ayarla; her protokol katmanında aynı şekilde bulunur ve o protokol katmanının kendi yapılandırma dosyasına kaydedilen ayarla eşzamanlı tutulur.
GET /api/nics	Ağ kartlarını listele, seçili olanı göster.
POST /api/nics	Bir ağ kartı seç (MAC ile, isteğe bağlı doğrulamalı).
GET /api/health	Basit canlılık kontrolü — {"status":"ok"}.
GET/POST/DELETE /api/license	Lisans durumunu oku, bir anahtarı etkinleştir ya da kaldır. Tek bir lisans beş protokol katmanının tümünü kapsar (§12.3).
GET /api/logs?after={id}	Verilen kimlikten daha yeni servis günlüğü kayıtları.
GET /api/paths	Diskte yapılandırma ve günlük dosyası yolları (§B).
GET /api/devices/{name}/tags PUT /api/devices/{name}/tags POST /api/devices/{name}/tags DELETE /api/devices/{name}/tags/{tagName} GET /api/devices/{name}/tags/values POST /api/devices/{name}/tags/write	Etiketleri listele / tümüyle değiştir / ekle-ya-da-güncelle / kaldır; güncel değerleri oku; bir ya da daha fazla Çıkış yönündeki etikete yaz (§5.3).

Yukarıdaki /api/logs ve /api/paths ile ilgili not: bunları o çıplak yollarda yalnızca PROFINET Controller ve PROFINET Device sunar. EtherCAT Master ve her iki EtherNet/IP rolü aynı bilgiyi bunun yerine protokol katmanı önekli bir yolda sunar — EtherCAT Master'da /api/ethercat/logs ve /api/ethercat/paths (§A.2), hem EtherNet/IP

Scanner hem Adapter'da `/api/ethernetip/logs` ve `/api/ethernetip/paths` (§A.3, §A.4). PROFINET Device ayrıca hiç `/api/paths` ucu sunmaz — yalnızca `/api/logs` vardır.

A.2 EtherCAT Master — cihazlar

Bu bölümdeki her yol `/api/ethercat/` öneki altındadır — EtherCAT protokol katmanı jenerik `/api/devices` biçimini kullanmaz.

Uç nokta	Amaç
GET <code>/api/ethercat/status</code> GET <code>/api/ethercat/bus</code>	Protokol katmanı durum özeti (çalışıyor/durduruldu, mesaj), ya da tam veriyolu görüntüsü (NIC, slave'ler, çevrim süresi).
POST <code>/api/ethercat/start</code> POST <code>/api/ethercat/stop</code>	EtherCAT veriyolu/slave haberleşmesini elle başlat ya da durdur — web arayüzü durdurmadan önce onay ister (§6).
GET <code>/api/ethercat/nics</code> GET <code>/api/ethercat/nic</code> POST <code>/api/ethercat/nic</code>	Kullanılabilir ağ kartlarını listele; halkanın hangisini kullandığını oku ya da seç (ad ile, isteğe bağlı MAC geçersiz kılma ile) (§6).
POST <code>/api/ethercat/bus/settings</code>	Cyclic alışveriş aralığını (çevrim süresi, milisaniye) ayarla (§6).
GET <code>/api/ethercat/slaves</code> PUT <code>/api/ethercat/slaves</code> DELETE <code>/api/ethercat/slaves/{id}</code>	Yapılandırılmış slave'leri listele, birini ekle-ya-da-güncelle (upsert), ya da yapılandırmadan kaldır.
GET <code>/api/ethercat/slaves/status</code>	Slave başına canlı durum: yapılandırılmış her slave için ESM durumu ve istatistikler.
GET <code>/api/ethercat/slaves/{id}/input</code> POST <code>/api/ethercat/slaves/{id}/output</code>	Slave'in anki ham giriş PDO baytlarını oku (hex), ya da çıkış PDO baytlarını yaz.
GET <code>/api/ethercat/slaves/{id}/tags</code> PUT <code>/api/ethercat/slaves/{id}/tags</code> POST <code>/api/ethercat/slaves/{id}/tags</code> DELETE <code>/api/ethercat/slaves/{id}/tags/{tagName}</code> PUT <code>/api/ethercat/slaves/{id}/tags/{tagName}</code> GET <code>/api/ethercat/slaves/{id}/tags/values</code> POST <code>/api/ethercat/slaves/{id}/tags/write</code>	Bir slave'in etiketlerini listele / tümüyle değiştir / ekle-ya-da-güncelle; bir etiketi kaldır ya da yeniden adlandırıp güncelle; güncel değerleri oku; bir ya da daha fazla Çıkış yönündeki etikete yaz (§5.3).
POST <code>/api/ethercat/slaves/{id}/sdo/read</code> POST <code>/api/ethercat/slaves/{id}/sdo/write</code>	Bir CoE nesne sözlüğü girdisini (index/alt-index, hex veri) oku ya da yaz — başlangıç parametreleri ve tanılama için kullanılan SDO aracı (§6).
GET <code>/api/ethercat/esi-catalog</code> POST <code>/api/ethercat/esi-catalog/reload</code> POST <code>/api/ethercat/esi-catalog/directory</code> POST <code>/api/ethercat/esi-catalog/browse-folder</code>	İzlenen ESI klasörünü, indekslenen imza sayısını ve son yeniden yükleme zamanını bildir; yeniden taramayı zorla; izlenen klasörü değiştir; bir klasör seçici aç (yalnızca etkileşimli masaüstü oturumları) — ESI zinciri kurucusu (§6).
GET <code>/api/ethercat/logs</code> GET <code>/api/ethercat/paths</code>	§A.1'de anlatılan paylaşılan <code>/api/logs</code> ve <code>/api/paths</code> 'in, ESI katalog klasörünü de içeren, protokol katmanı öneki karşılıkları.

A.3 EtherNet/IP Scanner — cihazlar

Bu bölümdeki her yol `/api/ethernetip/scanner/` öneki altındadır (Target yönetimi, bağlantılar, IO) ya da `/api/ethernetip/` altındadır (keşif, EDS katalogu) — Scanner protokol katmanı jenerik `/api/devices` biçimini kullanmaz.

Uç nokta	Amaç
GET <code>/api/ethernetip/status</code>	Rol kimliği ve durum mesajı ("EtherNet/IP Scanner", çalışıyor).
GET <code>/api/ethernetip/scanner/targets</code> PUT <code>/api/ethernetip/scanner/targets</code> DELETE <code>/api/ethernetip/scanner/targets/{id}</code>	Yapılandırılmış Target'ları listele; birini ekle-ya-da-güncelle (IP, T→O/O→T assembly örnekleri ve boyutları, RPI — §7.2); birini kaldır, bu önce Forward_Close gönderir.
GET <code>/api/ethernetip/scanner/connections</code> GET <code>/api/ethernetip/scanner/engine-status</code>	Target başına bağlantı durumu, RPI, assembly boyutları ve istatistikler; ya da protokol katmanının tümünün etkin/devre dışı bayrağı (tek bir Target'ın bağlantısından ayrı).
POST <code>/api/ethernetip/scanner/stop</code> POST <code>/api/ethernetip/scanner/start</code>	Yapılandırılmış tüm Target'larla haberleşmeyi elle durdur ya da başlat.
GET <code>/api/ethernetip/scanner/targets/{id}/input</code> POST <code>/api/ethernetip/scanner/targets/{id}/output</code>	Target'ın anki O→T assembly baytlarını oku (hex), ya da T→O assembly baytlarını yaz.
POST <code>/api/ethernetip/scanner/discover?seconds={n}</code>	0.5-10 sn boyunca UDP ListIdentity yayını yap; bulunan her cihaz, mümkünse "Target olarak ekle" formunu ön-doldurmak için EDS katalogu ile eşleştirilir.
GET <code>/api/ethernetip/eds-catalog</code> POST <code>/api/ethernetip/eds-catalog/reload</code> POST <code>/api/ethernetip/eds-catalog/directory</code> POST <code>/api/ethernetip/eds-catalog/browse-folder</code> POST <code>/api/ethernetip/eds/parse</code>	Katalog klasörünün durumu; yeniden taramayı zorla; izlenen klasörü değiştir; bir klasör seçici aç (yalnızca etkileşimli masaüstü oturumları); ya da tek bir yüklenen .eds dosyasının assembly'lerini doğrudan ayırıştır (katalog klasörü kullanılmaz).
GET <code>/api/ethernetip/scanner/targets/{id}/tags</code> PUT <code>/api/ethernetip/scanner/targets/{id}/tags</code> POST <code>/api/ethernetip/scanner/targets/{id}/tags</code> DELETE <code>/api/ethernetip/scanner/targets/{id}/tags/{tagName}</code> PUT <code>/api/ethernetip/scanner/targets/{id}/tags/{tagName}</code> GET <code>/api/ethernetip/scanner/targets/{id}/tags/values</code> POST <code>/api/ethernetip/scanner/targets/{id}/tags/write</code>	Bir Target'ın etiketlerini listele / tümüyle değiştir / ekle-ya-da-güncelle; bir etiketi kaldır ya da yeniden adlandırıp güncelle; güncel değerleri oku; bir ya da daha fazla Çıkış yönündeki etikete yaz (§5.3).

A.4 EtherNet/IP Adapter — assembly'ler

Bu bölümdeki her yol `/api/ethernetip/adapter/` öneki altındadır. Scanner'ın aksine, Adapter'ın bağlı Scanner'ları listeleyen bir ucu yoktur — tam olarak tek bir Giriş/Çıkış/Config assembly kümesi vardır, bu yüzden kendi durumu (çalışıyor/durduruldu, son etkinlik) bunun yerine `/api/ethernetip/adapter/status`'tan okunur.

Uç nokta	Amaç
GET <code>/api/ethernetip/status</code> GET <code>/api/ethernetip/adapter/status</code>	Rol kimliği/durum mesajı, ya da Adapter'e özgü durum (şu anda bağlı bir Scanner olup olmadığı, son etkinlik).
GET <code>/api/ethernetip/adapter/config</code> PUT <code>/api/ethernetip/adapter/config</code>	Bu protokol katmanının bir Scanner'a sunduğu yerel Giriş/Çıkış/Config assembly örneklerini ve boyutlarını oku, ya da ayarla (§8.2).

Uç nokta	Amaç
POST /api/ethernetip/adapter/stop POST /api/ethernetip/adapter/start	Bir Scanner'ın bağlantısını kabul eden TCP/UDP dinleyicilerini elle durdur ya da başlat.
GET /api/ethernetip/adapter/output POST /api/ethernetip/adapter/input	Bir Scanner'dan en son alınan Çıkış assembly baytlarını oku, ya da Giriş assembly baytlarını yaz (bağlı gerçek bir Scanner olmadan tezgah testi).
GET /api/ethernetip/adapter/tags PUT /api/ethernetip/adapter/tags POST /api/ethernetip/adapter/tags DELETE /api/ethernetip/adapter/tags/{tagName} PUT /api/ethernetip/adapter/tags/{tagName} GET /api/ethernetip/adapter/tags/values POST /api/ethernetip/adapter/tags/write	Bu Adapter'ın (tek) etiket kümesini listele / tümüyle değiştir / ekle-ya-da-güncelle; bir etiketi kaldır ya da yeniden adlandırıp güncelle; güncel değerleri oku; bir ya da daha fazla Çıkış yönündeki (T→O) etikete yaz (§5.3).

A.5 PROFINET Controller — cihazlar

Uç nokta	Amaç
GET /api/devices	Tüm IO-Device'leri özet durumla listele.
GET /api/devices/{name}	Tam ayrıntı: slotlar, alt slotlar, istatistikler.
POST /api/stop POST /api/start	PROFINET protokol katmanını tüm IO-Device'ler için birden elle durdur ya da başlat — web arayüzü durdurmadan önce onay ister (§9.6).
POST /api/ip	Controller'ın kendi istasyon adı ve IP/alt ağ/ağ geçidi ayarlarını uygula; etkili olması için PROFINET katmanını yeniden başlatır.
GET /api/devices/{name}/io POST /api/devices/{name}/io	Alt modül başına o anki ham IO baytları (hex); bir çıkış alt modülüne yaz.
POST /api/devices/{name}/enabled	Bir cihazı etkinleştir/devre dışı bırak.
DELETE /api/devices/{name}	Bir cihazı yapılandırmadan kaldır.
POST /api/devices/xdb	Bir .xdb cihaz vekil dosyasını içe aktar (multipart/form-data) — içerdiği tüm cihazları tek adımda ekler (§9.3).
POST /api/devices/discover	DCP ağ taraması — bulunan her PROFINET istasyonunu, varsa otomatik GSDML katalog eşleşmesiyle birlikte listeler (§9.10).
GET /api/gsdml-catalog GET /api/gsdml-catalog/devices GET /api/gsdml-catalog/devices/{vendorId}/{deviceId}/{dapId}/modules	Katalog klasörünün durumu (dosya/cihaz sayısı, son tarama, hatalar); katalogdaki tüm DAP'ları listele; bir DAP'ın takılabilir modüllerini listele (§9.10).
POST /api/gsdml-catalog/reload POST /api/gsdml-catalog/directory POST /api/gsdml-catalog/browse-folder	Katalog klasörünü şimdi yeniden tara; taranan klasörü değiştir; bir klasör seçici aç (yalnızca etkileşimli masaüstü oturumları) (§9.10).
POST /api/devices/gsdml	Bir GSDML katalog girdisinden (DAP + slot/modül atamaları) doğrudan yeni bir cihaz kur ve ekle — hiç .xdb kullanılmadan (§9.10).

A.6 PROFINET Device — kendi yapılandırması

Uç nokta	Amaç
GET /api/device GET /api/slots	Bu Device'ın kendi kimliği (istasyon adı, Vendor/Device ID) ve anki slot/alt slot düzeni.
GET /api/modules POST /api/modules	Bu Device'ın kendi slot/alt slot modül düzenini oku ya da ayarla — IO-Controller'a (GSDML üzerinden) tanımlananla ve burada fiziksel olarak simüle edilenle eşleşmelidir (§10.3).
GET /api/io POST /api/io	Controller'ın gördüğü şekliyle Device'ın anki giriş baytlarını oku, ya da yaz (bir sensörü simüle ederek); Çıkış baytları Controller'dan gelir ve burada salt okunurdur. GET yanıtı, anki AR aşamasını ve varsa bağlı Controller'ın istasyon adını da taşır — ayrı bir /api/ar ucu yoktur.
GET /api/diagnostics POST /api/diagnostics/test-pattern POST/DELETE /api/diagnostics/force DELETE /api/diagnostics/forces POST /api/diagnostics/provider-status POST /api/diagnostics/reset-statistics	Tanılama kartının durumunu oku; bir slotta yinelenen bir test deseni sür; tek bir bayt aralığını ya da bir etiketin simüle edilen değerini zorla (ya da temizle); tüm etkin zorlamaları tek seferde temizle; bir modül arızasını simüle et (IOPS = Bad) ya da temizle; çerçeve/hata sayaçlarını sıfırla (§10).

Ek B

Yapılandırma Dosyaları

Her protokol katmanı, kendi Config\ klasörü altında, servis çalıştırılabilir dosyasının yanında, düz ve insan tarafından okunabilir JSON olarak yapılandırmasını tutar. Dosyalar normalde web arayüzünden düzenlenir; servis durdurulmuşken elle düzenlemek de işe yarar, ama servis çalışıyorsa elle yapılan değişikliklerin etkili olması için yeniden başlatılması gerekir.

B.1 Config\enginesettings.json

Host seviyesinde küçük bir dosyadır ve yalnızca tek bir şeyi tutar: beş protokol katmanından hangisinin aktif olduğu. Üst çubuktan ya da protokol katmanı seçiciden (§4) protokol katmanı değiştirildiğinde yazılır; servis bunu başlangıçta bir kez okur ve bir değişiklikten sonra birkaç saniye içinde yeni seçilen protokol katmanıyla kendini yeniden başlatır (§4). NIC seçimi, istasyon adı, IP/alt ağ/ağ geçidi ya da başka herhangi bir protokol katmanına özgü ayarı TUTMAZ — bunlar, aşağıdaki §B.2'de listelenen her protokol katmanının kendi ayar dosyasında bulunur.

B.2 Protokol katmanı başına ayarlar ve cihaz/Target listesi

Her protokol katmanı, kendi kimlik/ağ ayarlarını ve birden fazla uzak istasyonu yönettiği durumlarda cihaz/slave/Target listesini, paylaşılan jenerik bir ad yerine o protokol katmanına özgü bir dosya adı altında tutar. Birden çok istasyonlu protokol katmanlarında liste, her istasyonun adresini ve IO biçimini taşır (PROFINET Controller cihazları normalde bir .xdb dosyası içe aktararak doldurulur, §9.3, ve istasyon adını, IP'sini ve tam slot/alt slot düzenini taşır; EtherCAT Master slave'leri ESI ile eşleşen PDO haritalarını taşır, §6.2; EtherNet/IP Scanner Target'ları IP, assembly örnekleri/boyutları ve RPI taşır, §7.2). PROFINET Device ve EtherNet/IP Adapter tek bir istasyonu temsil eder, bu yüzden ayrı liste dosyaları yoktur — tek ayar dosyaları hem kimliği hem IO biçimini kapsar.

Protokol Katmanı	Ayar dosyası	Liste (yalnızca çoklu istasyon)
PROFINET Controller	Config\controllerssettings.json	Config\devices.json
PROFINET Device	Config\deviceconfig.json	— (tek istasyon)
EtherNet/IP Scanner	Config\ethernetip-scanner-targets.json (Target listesini de tutar)	aynı dosya
EtherNet/IP Adapter	Config\ethernetip-adapter-config.json	— (tek istasyon)
EtherCAT Master	Config\ethercat-master-config.json (slave listesini de tutar)	aynı dosya

Üst çubuktan ve her protokol katmanının kendi ayarlar/cihazlar kartlarından düzenlenir; servis durdurulmuşken elle yapılan bir değişikliğin uygulanması için servisin yeniden başlatılması gerekir.

B.3 Protokol katmanı başına etiket dosyası

Etiket tanımları (§5.3) da aktif protokol katmanına özgü bir dosya adında bulunur — {"kapsam": [etiketler], ...} biçiminde bir sözlük; "kapsam", çoklu istasyonlu protokol katmanları için bir cihaz/slave/Target adı, PROFINET Device ve EtherNet/IP Adapter için ise tek, sabit bir anahtardır. Web arayüzünün Etiketler kartından ya da .NET istemcisinin etiket metotlarından düzenlenir; çoklu istasyonlu bir protokol katmanının listesinden bir cihaz/Target kaldırmak buradaki etiketlerini de otomatik siler.

Protokol Katmanı	Etiket dosyası
PROFINET Controller	Config\tags.json
PROFINET Device	Config\tags.json
EtherNet/IP Scanner	Config\ethernetip-scanner-tags.json
EtherNet/IP Adapter	Config\ethernetip-adapter-tags.json
EtherCAT Master	Config\ethercat-master-tags.json

PROFINET Controller ve PROFINET Device, ilgisiz bir nedenle aynı dosya adını (Config\tags.json) kullanır — aynı anda yalnızca bir protokol katmanı çalıştığından, iki dosya asla bir arada bulunmaz.

B.4 Diğer dosyalar

Config\lisans.json etkinleştirilmiş lisansı tutar (uninstall.bat -RemoveConfig ile ya da lisansı web arayüzünden kaldırarak silinir). Tek bir lisans beşinin tümünü kapsadığından (§12.3), her protokol katmanında aynı dosya biçimi ve konum geleneği kullanılır. Dosya adının license.json değil lisans.json olduğuna dikkat edilmelidir. logs\ klasörü, Servis Günlüğü kartının canlı akışıyla aynı içeriği taşıyan dönen günlük dosyalarını tutar. Bir cihaz kataloğuna sahip protokol katmanları bunu kendi alt klasöründe de tutar — Config\profinet-controller-gsdml\ (GSDML), Config\ethercat-master-esi\ (ESI), Config\ethernetip-scanner-eds\ (EDS) — o protokol katmanının /api/paths ucunda bildirilir (§A).

Ek C

Sözlük

C.1 PROFINET

Terim	Tanım
AR	Application Relation — Controller ile bir IO-Device arasındaki mantıksal bağlantı.
IOCR	IO Communication Relation — bir AR'ın parçası olarak kurulan cyclic veri kanalı.
DCP	Discovery and Configuration Protocol — bir cihazı istasyon adıyla bulmak ve gerekirse ad/IP atamak için kullanılır.
RPC	PROFINET'in gerçek bağlantı-yönetim protokolü, UDP/IP üzerinden (port 34964) — Connect, Write/PrmEnd ve Release için kullanılır.
DAP	Device Access Point — bir cihazın GSDML'sindeki DeviceAccessPointItem; cihazın kendisini temsil eder. Her zaman Slot 0'dır, sabittir (kaldırılamaz ya da değiştirilemez) ve GSDML'de tanımlı takılabilir modülleri cihazın diğer slotlarını doldurur (§9.10).
IOPS / IOCS	IO Provider Status / IO Consumer Status — her yönde her alt modülün cyclic verisine eklenen birer durum baytı: IOPS, üreticinin az önce gönderdiği verinin iyi mi kötü mü olduğunu söyler, IOCS ise tüketicinin bunu gerçekten kullanıp kullanmadığını söyler. Device tanımlama kartındaki "modül arızası simüle et", bir alt modülün IOPS'unu Bad'e zorlar (§10).
PrmEnd	Parametreleme aşamasını bitiren ve cihazdan yapılandırmasını uygulamaya başlamasını isteyen RPC çağırısı.
GSDML	General Station Description Markup Language — bir cihazın üreticisi tarafından yayınlanan, yeteneklerini, modüllerini ve parametrelerini tanımlayan XML dosyası.
XDB	Cihaz vekil dosyası — TIA Portal'ın bir Controller PC istasyonu proje yapılandırmasının derlenmiş dışı aktarımı; Controller bunu, bir cihazın ham GSDML'si yerine doğrudan okur (§9.5).
IO-Controller	Master istasyon — bu ürünün, PROFINET Controller protokol katmanını çalıştıran hali.
IO-Device	Modüler I/O'ya sahip uzak bir istasyon — ya gerçek saha donanımı, ya da PROFINET Device protokol katmanını çalıştıran bu ürün.
Watchdog	Bir Controller'in cyclic bir çerçeve gelmeden bekleyip AR'ı kayıp ilan ettiği süre; güncelleme süresi / zaman aşımı çarpanı olarak da adlandırılır.

C.2 EtherCAT

Terim	Tanım
ESI	EtherCAT Slave Information — bir slave'in üreticisi tarafından yayınlanan, PDO'larını, parametrelerini ve desteklenen modlarını tanımlayan XML dosyası; her slave'in SII imzasına karşı otomatik eşleştirilir (§6.2).
SII	Slave Information Interface — her slave'in taşıdığı, Üretici Kimliği'ni, Ürün Kodu'nu ve Revizyon'unu halkanın kendisi üzerinden bildiren küçük bir EEPROM.
PDO	Process Data Object — bir slave'in her çevrimde alışverişte bulunduğu cyclic giriş/çıkış bit/bayt düzeni; eşleşen ESI girişinden okunur.

Terim	Tanım
SDO	Service Data Object — bir slave'in CoE nesne sözlüğüne acyclic mailbox okuma/yazma; başlangıç parametreleri ve tanılama için kullanılır.
ESM	EtherCAT State Machine — her slave'in cyclic veri akışa başlamadan önce yürüdüğü INIT → PREOP → SAFEOP → OP dizisi.
SM	Sync Manager — Master'ın çerçevesi ile slave'in uygulaması arasında cyclic ve mailbox veriyi aktaran slave tarafı tampon mekanizması.
FMMU	Fieldbus Memory Management Unit — bir slave'in yerel fiziksel belleğini Master'ın tek mantıksal process image'ına eşler.
CoE	CANopen over EtherCAT — EtherCAT'in SDO parametreleri ve tanılama için CANopen'dan yeniden kullandığı nesne sözlüğü kuralı.

C.3 EtherNet/IP

Terim	Tanım
CIP	Common Industrial Protocol — EtherNet/IP'nin TCP (explicit mesajlar) ve UDP (cyclic IO) üzerinden taşıdığı nesne yönelimli mesajlaşma modeli.
Forward Open	Bir Scanner ile bir Target arasında Class 1 (cyclic) bağlantı kuran, RPI ve assembly boyutlarını görüşen CIP servisi.
Forward_Close	Bir Class 1 bağlantısını sonlandıran CIP servisi. Bu protokol katmanı, bir Target'ın cyclic alışveriş döngüsü her sona erdiğinde bunu gönderir — Target, Scanner'ın listesinden silinirken, kaldırılmadan hemen önce de dahil (§A.3).
Class 1 / Class 3	Amaca göre CIP bağlantı türleri: Class 1, bir Forward Open'in IO assembly'leri için kurduğu cyclic, bağlantılı UDP trafiğidir; Class 3 ise EDS ayrıştırma ya da cihaz kimliği okuma gibi tek seferlik okuma/yazmalar için kullanılan bağlantılı, istek/yanıt TCP mesajlaşmasıdır (explicit mesajlar).
RPI	Requested Packet Interval — bir Scanner'ın bir Target'tan çalışmasını istediği cyclic güncelleme hızı.
Originator / Scanner	Bağlantıyı açan ve alışveriş süren istasyon — bu ürünün EtherNet/IP Scanner protokol katmanı, gerçek bir sistemde tipik olarak bir PLC.
Target / Adapter	Bir Forward Open kabul edip assembly'ler sunan istasyon — bu ürünün EtherNet/IP Adapter protokol katmanı, gerçek bir sistemde tipik olarak bir saha cihazı.
Assembly	CIP'in tek birim olarak alışverişte bulunduğu adlandırılmış bir bayt grubu — Giriş, Çıkış ya da Config, her biri kendi örnek numarası ve boyutuyla.
ListIdentity	CIP/UDP yayın keşif komutu (port 44818) — PROFINET'in DCP Identify'nin EtherNet/IP karşılığı. Scanner'ın keşif özelliği bunu yayınlar ve yanıt veren her cihazın Vendor ID, Product Code, Device Type ve Revizyon'unu geri okur, ardından bu kimliği EDS katalogu ile eşleştirir (§A.3).
EDS	Electronic Data Sheet — bir Target'ın üreticisi tarafından yayınlanan, assembly'lerini ve parametrelerini tanımlayan dosya biçimi (GSDML/ESI'nin EtherNet/IP karşılığı).

C.4 Ortak terimler

Terim	Tanım
Protokol Katmanı	Bu ürünün sunduğu beş protokol uygulamasından biri — EtherCAT Master, EtherNet/IP Scanner, EtherNet/IP Adapter, PROFINET Controller, PROFINET Device. Aynı anda yalnızca biri çalışır.
Tag	Bir cihazın ham IO verisi içindeki bir bayt/bit aralığına verilen ad — böylece ham offset yerine web arayüzünden ya da .NET istemcisinden adıyla okunabilir/yazılabilir (§5.3).
Cycle time	Bir protokol katmanının cyclic alışverişini sürdüğü aralık — protokol katmanı başına yapılandırılabilir, tipik olarak tek haneli milisaniyeler.
Jitter	Bir çevrimin gerçek zamanlamasının yapılandırılmış çevrim süresi/RPI/güncelleme süresinden ne kadar saptığı; Bölüm 13'te anlatılan gerçek zamanlı ince ayarla milisaniyenin altında tutulur.
Machine ID	Lisans kartında gösterilen ve bu PC'de bir anahtarı etkinleştirmek için kullanılan kimlik. Üç bağımsız donanım/işletim sistemi kaynağından ikisine dayanır (SMBIOS sistem UUID'si, Windows kurulum GUID'i, sistem sürücüsü birim seri numarası); her biri tuzlanıp özetlenir, böylece ham değerler makineden hiç çıkmaz. Üçünden en az ikisi eşleştiği sürece lisans geçerli kalır, bu yüzden olağan bakım (disk değişimi, ağ kartı değişimi, işletim sistemi yeniden kurulumu) lisansı sessizce bozmaz (§12.3).
Trial period	Bir lisans anahtarı gerekene kadar geçerli 30 günlük deneme penceresi. Başlangıç tarihi üç konumda (kurulum klasörü, %ProgramData% ve registry) yedekli olarak kaydedilir; her kayıt bu PC'nin Machine ID'sine HMAC ile mühürlenir, böylece süreyi uzatmak için elle düzenlenemez; okumada üçü arasındaki en eski kaydedilen başlangıç ve en ileri "son görülme" zamanı geçerli sayılır, sistem saatinin geriye alınması ise süresi dolmuş bir deneme olarak değerlendirilir (§12.3).