EDGE-Skills

Developer Training

Trainer Script

Trainer Script: Companion material for the presenter

34 slides · vs10.5 · July 2026

This script accompanies the trainer slide by slide with background knowledge, context, and talking points that go beyond the keywords shown on the slides.

Co-funded by
the European Union

Slide 1

Title Slide: EDGE-Skills Developer Training

Welcome to the EDGE-Skills Developer Training. Over the next 90 minutes, we cover what you actually build, deploy, and integrate when onboarding an offering into the EDGE-Skills data space. We work with the Prometheus-X open source stack.

The focus is on architecture, protocols, and concrete implementation, not on business logic or governance strategy. Every concept is demonstrated through a real use case.

Slide 2

Overview: What This Training Is About

Four guiding principles for this training. Architecture-first: we walk through the complete technical stack layer by layer. Developer-focused: what you concretely build, deploy, and integrate. Protocol-Driven Thinking: the Prometheus-X open source stack with its consent-driven PDI approach, implementing DSP for interoperability and extending it for personal data sharing.

Important note: DSP and DCP are in the ISO standardisation process. Hands-on reference case: we trace a real data sharing scenario through every component. This is not a management overview; this is a technical deep dive for developers.

Slide 3

Foundation: What Is a Data Space?

The definition is based on the DSSC Blueprint (Data Spaces Support Centre) and the emerging ISO/IEC 20151 standard. Seven essential elements: environment rather than platform, trust through verifiable credentials, peer-to-peer via connectors, a governance authority that maintains rules, DSP and DCP protocols plus the PDI (Personal Data Intermediary), machine-readable ODRL policies, and supporting services.

What a data space is NOT: not a data lake, not a central platform, not an API gateway, not blockchain-dependent, not a centralised identity provider. The key shift for developers: you build components in a protocol-driven ecosystem, not a monolithic application.

Your architecture decisions must be interoperable.

Slide 4

Why This Matters for Developers

Four paradigm shifts that change your thinking as a developer. From single application to decentralised protocol ecosystem: no longer a single app but distributed components communicating via standard protocols. From central auth with OAuth or API keys to verifiable credentials and ODRL policies; identity is decentralised, no central auth server.

From data through a central hub to direct peer-to-peer data plane transfer: data flows directly between participants, no central broker. From API integration to working with the Prometheus-X open source stack: instead of REST calls against a single API, you configure building blocks and consent flows.

Slide 5

The Prometheus-X Architecture: Five Building Blocks

Five building blocks, consent-driven PDI: that is the Prometheus-X open source stack. The PDC Connector for peer-to-peer data exchange, Data Planes where data actually lives, the Identity Hub for credential management, the Catalogue for federated discovery, and the Contract Service for automated agreements.

Prometheus-X implements DSP as an interoperability protocol for discovery, negotiation, and transfer. DCP handles credential-based identity verification. The key differentiator is the PDI (Personal Data Intermediary), enabling consent-driven personal data sharing where individuals manage their consents from a single place.

The five components: connector (PDC), data planes, identity hub, catalogue, contract service.

Slide 6

Critical Distinction: Implementation ≠ Deployment

Deploying containers is NOT the same as implementing a data space. Five things must be fulfilled: the PDC Connector implements DSP correctly for interoperability (not just reachable, but protocol-conformant). The identity hub presents valid credentials (not just configured, but cryptographically verifiable).

The data plane handles actual data transfer (not just passing a health check). Policies are evaluable at runtime (not just defined, but enforceable). The participant is provisioned with credentials (not just registered). This is the most common confusion in data space projects.

Slide 7

Think in Protocols, Not in APIs

The Prometheus-X approach in detail. DSP (Dataspace Protocol) is implemented as an interoperability protocol: catalogue discovery where the consumer queries the provider's offerings, contract negotiation through the state machine REQUESTED, OFFERED, AGREED, VERIFIED, FINALIZED.

Transfer coordination where the control plane orchestrates data plane activation. All via HTTPS and JSON-LD, machine-to-machine, zero-trust. DCP (Decentralised Claims Protocol) handles peer-to-peer credential exchange for cross-data-space identity interoperability.

Credentials come from the identity hub and are cryptographically verified. The PDI (Personal Data Intermediary) is Prometheus-X's key differentiator, enabling consent-driven personal data sharing where individuals centrally manage, audit, and revoke permissions. The Service Chain Protocol enables multi-participant data flows orchestrated across organisations. Using the Orchestrator component (available from PDC v1.9), participants can chain Data Offers, Service Offers, and Infrastructure Offers into visual processing pipelines. Use-case logic is defined per Data Space, allowing flexible composition of analytics, transformation, and enrichment services.

Slide 8

The Four Layers Developers Must Understand

Four technical layers you must master as a developer. First, the control plane: the PDC with catalogues, negotiations, contracts, and ODRL policy evaluation, implementing DSP for interoperability. This is the brain. Second, the data plane: actual data transfer via pull, push, or stream.

REST APIs via PDC. This is the muscle. Third, the identity hub: DIDs using did:web, verifiable credentials, and credential proof composition for identity verification. Fourth, tenant management: multi-tenant lifecycle (and critically: NOT in the trust decision path).

Trust decisions happen directly between connectors peer-to-peer.

Slide 9

EDGE-Skills Use Case: Skill Gap Analytics

The EDGE-Skills project: an EU-funded data space for education and skills, led by Prometheus-X with 36 organisations across 8 EU countries. Our use case: Schülerkarriere shares anonymised student skill profiles, imc consumes them for skill gap analysis.

The data flow: student profiles with interests, competencies, and learning progress flow through an analytics service to personalised skill gap reports for students and employers. Why this use case: a real EDGE-Skills scenario combining personal data protection under GDPR, sovereign data sharing, and cross-organisational analytics, all the technical challenges at once.

Slide 10

Reference Case: Skill Gap Analytics for Students

The three hubs in detail. Hub A (provider): Schülerkarriere as student career platform, publishes anonymised skill profiles. Hub B (consumer): analytics service provider, consumes skills data for gap analysis. Hub C (end customer): students and employers, benefit from personalised skill gap reports.

The trust model: access is controlled by verifiable credentials plus GDPR consent. The policy: any organisation with valid membership and data processing credentials can access the skills data. No identity checks, no API keys, no manual approvals; everything automated through the protocol.

Slide 11

Step 1: The Connector (Control Plane)

The connector is the central software agent representing a participant in the data space. Four core functions: catalogue publication: a dynamic catalogue based on consumer credentials. Different consumers see different offerings.

Contract negotiation: verify credentials, evaluate policies, produce binding contract agreements. Policy evaluation: ODRL policies evaluated at runtime without redeployment. Transfer coordination: selects the data plane and issues authorisation signals.

Never handles data directly. Deployment options: standalone with one connector per organisation, or multi-tenant via PDC.

Slide 12

Step 2: Build and Register a Data Plane

The data plane is where actual data moves, and where you write custom code. Three transfer patterns: pull: consumer fetches from provider via APIs and queries. Push: provider sends to consumer via batch exports and files.

Stream: continuous flow for IoT and telemetry. Architecture principles: placement flexibility: operate the data plane where data lives. Protocol diversity: specialised data planes for different protocols. Independent scaling: no impact on policies.

Security boundaries: trust is separate from data. Integration via REST APIs through PDC, interoperable with EDC SDKs in Go, Java, Rust, and .NET. For simple file or JSON sharing over HTTP, you need minimal custom code.

Slide 13

Data Plane and Transfer Lifecycle (EDC Interoperability)

The four data plane lifecycle signals (EDC-based data planes): START: control plane initiates the transfer. SERVE: data plane serves data. SUSPEND: optional pause. TERMINATE: end transfer. Endpoint data references: after successful negotiation, the consumer receives an EDR containing endpoint URL and a time-limited access token.

The data plane validates the token before serving any data. Capability registration: data planes declare their supported source types and transfer types. The control plane routes transfers to the right data plane based on capabilities.

Deployment options: on-premises, VMs, cloud, or managed.

Slide 14

Step 3: Identity Hub & Decentralised Identity

The identity hub is each participant's wallet for credentials and DIDs. Four areas: DID management: creates did:web identifiers, manages signing keys, and publishes DID documents. Credential storage: membership credentials, attribute credentials, and role-specific credentials.

Proof composition: selects relevant credentials, creates cryptographic presentations for peer-to-peer identity verification without central authority. Verification: checks signatures, validity, revocation status, and trust anchors. The trust sequence: DID creation, credential acquisition, hub storage, proof presentation, verification, policy evaluation.

That is the complete identity lifecycle.

Slide 15

Verifiable Credentials & Trust Anchors

The central concept: trust is attribute-based, not identity-based. Credential structure: claims (what is attested). Issuer identity (who signed it). Cryptographic proofs (verifiable signature). Validity periods (expiration date).

Lifecycle: issuance, storage, presentation, verification, renewal, revocation. Trust frameworks such as Gaia-X and EBSI define which issuers and credential types are trusted. Five policy levels using credentials: segmentation (catalogue visibility by credential).

Access (data sharing requirements). Usage (post-transfer rules). Contract (sharing agreements). Trust (trust contexts).

Slide 16

Step 4: Define Policies with ODRL

Machine-readable rules evaluated at runtime. Three evaluation points: at catalogue discovery, access policies filter what consumers see. At contract negotiation, contract policies verify credentials. At transfer execution, usage policies govern data use.

ODRL control mechanisms in four tiers: broad access for all active members. Credential-based for specific certifications. Bilateral for a specific partner only. Framework-governed for members who accepted a particular agreement.

The key advantage: ODRL policies can be updated at runtime; change one policy and it takes immediate effect, no code change, no redeployment needed.

Slide 17

Step 5: Publish Data

Three control plane objects for publication. First: create asset: metadata including ID, type, and product plus DataAddress pointing to the data plane. The control plane holds only metadata, no data. Second: define policy: ODRL policy specifying access requirements with credential-based authorisation.

Third: contract definition: links asset and policy. Makes the offering discoverable and negotiable in the catalogue. The central principle: the actual data stays on the provider's data plane. The control plane holds only governance.

This is the core principle of data sovereignty.

Slide 18

Step 6: Consume Data

The three-step data consumption protocol. Catalogue discovery: query the provider’s catalogue endpoint. The catalogue is generated dynamically based on consumer credentials. Key characteristics: no portal accounts, no API keys, no custom code, no manual approvals, standardised across all providers.

Contract negotiation: REQUESTED, AGREED, VERIFIED, FINALIZED. Seconds, no human intervention. Transfer initiation: receive EDR with endpoint and token, retrieve data directly from the provider's data plane. For providing data: deploy your own data plane.

For consuming only: a standard connector is sufficient.

Slide 19

VisionsTrust: Your Prometheus-X Implementation Reference

VisionsTrust is the SaaS reference implementation of the Prometheus-X building blocks for discovery, contracts, consent, and data exchange. Three core services: the catalogue for discovery, onboarding, and configuration dashboards.

The contract service for negotiation, signatures, and policy management. The consent service for individual consent and data exchange triggers. As a developer, VisionsTrust is your concrete implementation reference; here you see how the abstract protocols are realised in a production platform.

Slide 20

VisionsTrust Architecture Overview

How the three services connect through the Dataspace Connector PDC. The catalogue: participant onboarding, offer and resource management, project discovery, marketplace browsing. The contract service: contract negotiation, policy evaluation, signature management, agreement lifecycle.

The consent service: consent records, privacy notices, user registration, exchange authorisation. Everything runs through the PDC as the central coordination layer. Important verification step: after PDC setup, check Settings, then Endpoints tab to confirm your HTTPS domain appears correctly.

Slide 21

Setting Up the Dataspace Connector (PDC)

The PDC enforces contracts, evaluates policies, manages consent, and communicates with the Prometheus-X services. Six configuration parameters: endpoint: your own HTTPS domain. catalogUri, contractUri, consentUri: the fixed VisionsTrust endpoints. serviceKey and secretKey: your unique keys from the VisionsTrust dashboard under Settings.

Auto-registration: once configured, your PDC automatically registers with VisionsTrust. Verify via Settings, then Endpoints. Optional: credentials can be generated through the PDC. Use credential identifiers in metadata, never the actual values.

Slide 22

The Complete Onboarding Flow

Two parallel paths: non-technical for the provider and technical for the developer. Provider side: register on VisionsTrust, create offer bundling resources as a catalogue product, add data and service resources with metadata, type, location, and sample, then publish to catalogue.

Developer side: deploy PDC, configure PDC for catalogue, contract, and consent, verify connectivity via ping under Settings then Endpoints, set up representations with source type, URL, and credentials, then test in Tech Space with test PDCs.

Both paths must be synchronised; this is the PM-developer interface.

Slide 23

Resources, Offers, and the Catalogue

Key distinction: a resource is an individual metadata unit. An offer bundles resources into a discoverable product. Resources alone never appear in the catalogue. Mandatory fields: resource name, resource type, description, data location, anonymised extract as JSON sample, and if applicable the personal data flag.

The personal data flag triggers consent requirements and requires the userId parameter in the endpoint. Technical representations in the tech tab: source type is HTTP REST, URL is the endpoint for data retrieval, security is none or API-key using credential identifiers, payload for API consumption flow configuration.

Slide 24

Projects, Contracts & Negotiation

No data exchange without a signed project contract. The flow: create project, invite providers or providers request to join, negotiate terms with modify and counter-offer, sign contract as mandatory for all participants, then exchange data with active offers.

The negotiation cycle in detail: orchestrator invites provider, provider reviews terms, either party can modify and counter, cycle repeats until agreement, provider signs contract and offers activate. Key terms: participant as entity with data space identifier, project as scoped ecosystem for data exchanges, orchestrator manages project and builds service chains, contract as data sharing agreement between members.

Slide 25

Four Data Exchange Protocols in Practice

Four exchange types for different scenarios. B2B non-personal: direct organisation-to-organisation data sharing, no consent needed, contract-based authorisation only, simplest exchange type. Consent-driven: individual consent required, PDI consent service involved, endpoint must include userId, resource flagged as personal data.

API consumption: data provider consumes services, resource marked as API payload, PDC adjusts exchange protocol, response sent back to provider. Service chains: multi-participant workflows, orchestrator builds chain visually, data plus service plus infrastructure offers, requires PDC v1.9.0+.

Slide 26

Testing with VisionsTrust Tech Space

Dedicated testing environment with VisionsTrust-controlled test PDCs. Four supported exchange tests: project exchange for one-to-one contract-based, service chain for multi-participant flows, API consumption per PDC protocol specification, consent-driven with personal data simulation.

Testing tools: ping connector for PDC connectivity, log viewer for real-time monitoring, CSV export for troubleshooting, custom parameters for exchange parameters with payload examples, exchange triggers from PDC v1.10.0+.

Common configuration errors: missing tech config: set REST source type and endpoint URL in data representation. Missing API flow: check resource API box. Missing PII declaration: enable personal data flag and add userId query parameter.

Slide 27

Operational Tooling for Data Spaces

Tools and capabilities for managing Prometheus-X data spaces in production. Eight areas: data catalogue for asset registration and classification. Data spaces for browsing federated spaces and participants. Connections for agreement negotiation and exchange management.

Transfers for monitoring data movement and status. Compliance for DPP, CSRD, AI Act, LkSG, EUDR, and DGA. Governance for classification, policy enforcement, and quality scores. Policy engine for multi-licence management and conflict resolution.

Graph view for network visualisation of the ecosystem. Data sources connected via PDC: HTTP/REST endpoints, file storage, cloud services.

Slide 28

Multi-Tenant Deployment with PDC

The PDC Tenant Manager for scalable operations. Important upfront: the PDC Tenant Manager is NOT in the trust decision path. Trust decisions happen directly between connectors peer-to-peer. Two-step provisioning: step 1: create tenant with POST including tenant-ID, display name, and description.

Step 2: deploy participant profile with DID binding and Dataspace Profiles. Four sequential activity agents: IDP agent for OAuth2 clients, PDC agent for participant contexts, registration agent as credential issuer, onboarding agent for VCs and storage.

Components: tenant manager, provision manager, NATS, and PostgreSQL.

Slide 29

Step 7: Validate the Full Flow End-to-End

Six validation steps for the end-to-end check. Membership: credential proves requirement fulfilment. Discovery: catalogue search for contract offers. Negotiation: credential verification and auto-negotiation. Agreement: contract agreement as authorisation artefact.

Execution: data transfer via pull, push, or stream. Monitoring: audit records for compliance. Validation checklist: can a consumer discover your offering in the catalogue? Are credentials correctly presented and verified during negotiation? Does the contract agreement get created successfully with status FINALIZED? Can data actually be transferred via the data plane with status COMPLETED? Are audit records generated for compliance?

Slide 30

Where Implementation Often Fails

Six typical failures, not edge cases but the most common reasons data space implementations fail. Deploying containers without configuring credentials. Forgetting to register the data plane. Writing policies that reference non-existent credential types.

Hardcoding trust instead of using the PDC protocol stack. Treating the catalogue as static documentation. Skipping governance: no trust framework, no credential issuance. Each of these points is avoidable if you take the architecture seriously and work through the checklist from slide 29.

Slide 31

DevOps & Operational Practices

Infrastructure components: PostgreSQL as database, NATS for messaging, Kubernetes for orchestration, secret store for credential management. Deployment principles: config-based isolation, single deployment for all tenants, shared identity provider.

Observability: health and readiness APIs, Prometheus metrics, log aggregation, alerting, and the connector observability API. The core principle: for file and JSON sharing over HTTP, the only custom code is a minimal data plane.

Invest in configuration, not reimplementation. The Prometheus-X building blocks handle the bulk of the work.

Slide 32

The Implementation Toolkit

Three pillars of your toolkit. Prometheus-X open source stack: catalogue, contract service, and consent service as core components. VisionsTrust as SaaS reference implementation for onboarding, discovery, negotiation, and exchange.

PDC as operational platform for catalogue, governance, compliance, and exchange. Interoperable ecosystem: compatible with EDC in Go, Java, Rust, and .NET plus SIMPL and FIWARE stacks. Trust frameworks: Gaia-X, EBSI, and DSSC for decentralised identity, credentials, and trust anchors.

Learning resources at prometheus-x.org for concepts, building blocks, and integration guides.

Slide 33

Exercise: Apply This to Your Implementation Context

Five questions for your own data space project. What data do you share or consume: define assets, metadata, DataAddress. What policies govern access: credential types, ODRL rules, policy levels. What data plane do you need: transfer pattern, wire protocol, integration approach.

What trust framework applies: DGA, issuer service, onboarding process. What is your deployment model: single or multi-tenant, infrastructure, observability. Take five minutes and sketch out answers. If you have a real project, use that.

Otherwise, take the skill gap use case as a starting point.

Slide 34

Key Takeaways: What to Remember as a Developer

Five key messages. First: Protocol-driven ecosystems, not monolithic apps: the Prometheus-X open source stack with its PDI and consent-driven approach is your foundation. Second: trust via verifiable credentials, not API keys; identity hub and ODRL policy evaluation replace central auth.

Third: connector is the brain, data plane is the muscle; never mix trust decisions and data movement. Fourth: configuration over custom code: the Prometheus-X building blocks handle 90 percent of the work. Fifth: validate end-to-end, not component by component; the full chain must work together.

Thank you, questions?