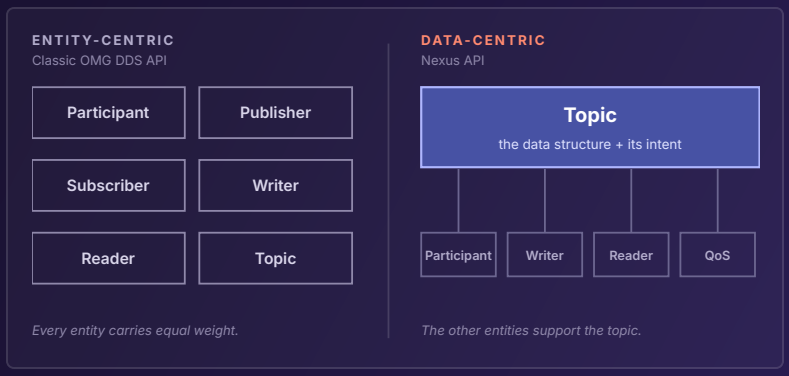
DATA-CENTRIC API FOR DDS

Nexus API

Put the data at the center. **The full capability of DDS, organized around the topic.**



THE DIFFERENCE The same DDS model, with a different center of gravity.

The OMG DDS API is **entity-centric**. Participant, publisher, subscriber, writer, reader, and topic all carry equal weight, and the data, which is what the application actually cares about, is one entity among many.

The Nexus API is **data-centric**. The topic, meaning the data structure and how that data behaves, is at the center. Participants, writers, readers, and QoS are all still part of the model; they support the topic instead of competing with it. Application code says what the data is and how it behaves, not how the infrastructure is assembled.

A complete DDS API, in a modern form

Nexus is not a wrapper. It is a native API to DDS, carrying the capability of the standard in a data-centric form. On the wire it is standard DDS, so Nexus applications interoperate with any compliant DDS system.

Fewer decisions, made in the right place

	ENTITY-CENTRIC: OMG DDS API
1	Create a domain participant
2	Register the type and create the topic
3	Create a publisher or subscriber
4	Create the writer or reader
5	Set QoS on each, compatible on both sides
6	Delete entities in the right order

	DATA-CENTRIC: NEXUS API
1	Name the topic
2	Say what kind of data it is
3	Join, then put or read
4	Leave when you're done

Designed for systems that have to be right

Intent-based QoS

State, event, or stream. Writers and readers get compatible QoS by construction, with overrides where needed.

Explicit lifecycle

Join, leave, and shut down when you say so. No hidden destructor work.

One binding per topic

One binding per domain and topic name, so conflicting definitions fail early.

Traceable configuration

Code, environment, or file, with the source of each policy recorded.

AT A GLANCE

Model
Data-centric: the topic is the primary object

Language
Modern C++ (namespace `databus`)

Data model
IDL, JSON, or XML, generated with `nexus_gen`

Build
CMake integration through `dds_target`

Standards
DDSI-RTPS and DDS-XTypes on the wire

Interoperability
Works with any standards-compliant DDS system

Status
In beta now; 1.0 in Q4 2026

EDITIONS

Community
Free, with no capacity limits

Professional
Additional tooling and features

Enterprise
For programs and organizations

BUILT ON THE NEXUS API

Nexus TMS. A free MIL-STD-3071 Tactical Microgrid Standard reference implementation. See the Nexus TMS brief.

Say what the data is. Nexus handles the rest.

Most DDS QoS decisions follow from one question: what kind of data is this? Nexus captures the answer as an intent, so writers and readers get the right, compatible settings every time.

INTENT	MEANS	EXAMPLES
`state`	"Here's how I am right now." Late joiners get the latest value.	Status, mode, configuration
`event`	"This happened." Every message counts.	Alarms, commands, requests and replies
`stream`	"Here's the next reading." Newest wins; losing one is fine.	Sensor readings, telemetry

vehicle_status.cpp — ILLUSTRATIVE

```cpp
#include "databus/databus.hpp"
#include "fleet_types.h"        // generated by nexus_gen

DATABUS_TYPE(fleet::VehicleStatus, vehicleId);

// Publish this vehicle's current status.
databus::topic<fleet::VehicleStatus> out(databus::writer,
    "VehicleStatus", databus::intent::state);
out.join();
out.put(status);

// Anyone else can watch it, and read the latest per vehicle.
databus::topic<fleet::VehicleStatus> in(databus::observer,
    "VehicleStatus", databus::intent::state);
in.join();
databus::sample<fleet::VehicleStatus> last = in.latest("ugv-07");

out.leave();
in.leave();
databus::shutdown();
```

THE SAME JOB

Publish and watch one topic

	OMG API	NEXUS
You create	8 entities	2 topics
QoS	Set by hand, matched on both sides	`intent`
Latest value	Look up, read, check, return loan	`latest()`
Shutdown	Delete in the right order	`leave()`

From data model to running code

1

Define the data

Write types in IDL, JSON, or XML, or reuse the IDL you already have.

2

Generate

`nexus_gen` produces C++ types and XTypes TypeObjects, wired into CMake with `dds_target`.

3

Build on topics

Name each topic, give it an intent, and join. It interoperates with any compliant DDS system.

FROM THE FIELD

19 vs 49

"With the OMG API, you learn DDS before you build your device. **With the Nexus API, you just build your device."**

Developer on a MIL-STD-3071 program. The same compiled program took 19 lines with Nexus and 49 with the classic OMG DDS C++ API. Ask us for both versions.

The DDS you know.
The API it should have had.

TALK TO US

www.federateddesigns.com
eva@federateddesigns.com

© 2026 Federated Designs. All product names and trademarks are the property of their respective owners.