DECEMBER 2019

IN THE PIPELINE

2020 OSDU F2F DATES

- ***Houston:** Developer Boot Camp March 4 and 5*
- ***Amsterdam:** F2F March 23-24 and Developer Bootcamp March 25*
- ***Houston:** Spring F2F April 20-23*
- ***Houston:** Summer F2F July 13-16*
- ***Houston:** Fall F2F October 12-15*

THE OPEN GROUP
OPEN SUBSURFACE DATA UNIVERSE™ FORUM

OVERVIEW

OSDU Release 2 and Release 3 are already making significant progress toward 2020 deployments, and we want to share with you the important design and execution our teams are delivering.

This edition of **In the Pipeline** focuses on how we're enabling core capabilities, and what it means for you.

1. **OSDU R2 vs. R3:** What is changing?
2. **Services**: 7, 12 and 70
3. Understanding **Generic APIs and Domain APIs**

OSDU R2 v R3: What is changing?

OSDU Release 2 and Release 3 architectural designs are quickly taking shape as Developer and Deployment ready releases. Read on to understand the different release types.

OSDU DEFINITION OF DONE

MVP: A Minimum viable OSDU product allows Operators and Developers to experience the OSDU platform, learn the concepts of the system and start preparing their organization to roll out a future production ready system.
Developer ready: A developer can deploy the full baseline for testing/running the services as part of CI/CD pipeline.
Deployment ready: Someone besides a developer can deploy a tested system in a consistent manner. This is the baseline that the Systems Integrators will likely take to begin operationalizing.
Operational ready: This is a deployed version that has full operational support (monitoring/alerting, business continuity, and incident management).

OSDU RELEASE 2 ARCHITECTURE

You already know that OSDU Release 2 expands the OSDU scope and functionality to include seismic data, but you may not know is that **Release 2 is a brand new code** base based on the Schlumberger OpenDES contribution that is common across Azure, AWS and GCP.

This will be a **Developer Ready** release that contains the core services from OpenDES necessary to recreate the OSDU R1 workflows with the addition of Bluware OpenVDS and ZGY Seismic support. This "common code" base is the foundation for all new development to ensure that we maintain functional parity across all cloud platforms.

OSDU RELEASE 3 ARCHITECTURE

The primary goal of Release 3 is create a **Deployment Ready** system based on Release 2 along with: several additional services, schemas and domains from OpenDES and the data definition subcommittee, architectural and functional enhancements to support Global Deployment, and Deployment automation that allows non-developers to deploy an instance. Visit our OSDU R3 Architecture wiki for more on authentication, entitlements, search behavior, etc.

Release 3 extends the Release 2 footprint with many additional services and capabilities:
- **Schema registration** to allow the runtime support for new data types
- **Flexible ingestion framework** supporting these different schemas
- **Optimized domain stores for logs and seismic** on demand via an API vs. downloading files
- The capability for **data enrichment within the data platform** (producing new – better data)
- **Global deployment:** single customer, multi-region

For more information about OSDU Releases, please visit our comprehensive wiki in GitLab.

If your organization is not a Member of the OSDU Forum and would like to get involved, email: membership@opengroup.org

Services: 7, 12 and 70

The OSDU Services are core services that are published to the public as open endpoints. In presentations, we have seen reference to the 7, the 12 and the 70 services. Let's detail what that means for you.

7 SERVICES

The 7 refers to the **core services** of the common code base that form the essence of the system on which all other services are built.

These **core services** include support for:

- **Defining new metadata supporting new types**
- **Storing and retrieving metadata**
- **Security and policy (legal compliance)**
- **Ingestion and search**

Today we are integration testing these core services using OSDU data on all three cloud platforms.

12 SERVICES

12 services refers to the **core services plus additional services necessary to complete the Release 2 workflows** by adding OSDU R1 compatible ingestion, delivery services, and seismic support. With the 12 services, **OSDU R2** will have a **common code** base that extends OSDU R1 capability.

70 SERVICES

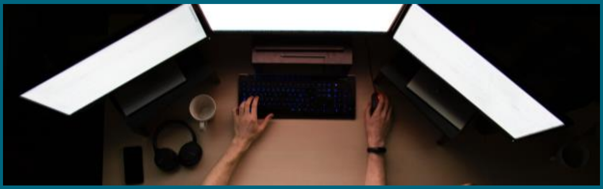
OSDU Release 3 expands scope, content, and capabilities with the completion of the OSDU and OpenDES integration. These include support for:

- **Better management of new data types**
- **Domain optimized APIs for bulk data (Seismic/Log)**
- **Enrichment framework (improve/transform)**
- **Frame of Reference services (Units, Spatial, …)**
- **New data types (backlog still being defined)**
- **New deployment models**

More information on the service roadmap can be found on the wiki or by contacting your PMC Leads Raj Kannan (Schlumberger) and Joe Nieten (AWS).

Understanding Generic APIs and Domain APIs

*APIs are the magic that helps us separate data from applications, but not all APIs are alike. OSDU was originally designed with data managers in mind, and **generic APIs** were a natural priority. However, as we evolve to add more capabilities to the platform, applications which require optimized access to data prefer **Domain APIs** for performance.*



APIs ARE THE ENGINE OF THE OSDU DATA PLATFORM

Generic APIs are the foundation that our OSDU data platform is built on. They are published as open endpoints to access metadata and data as generic types allowing both the continuous addition of new data types as well as supporting an ecosystem of:

- System developers extending the platform
- Systems integrators integrating the platform
- Customers adopting the platform

This was a logical starting point, but we are making changes in OSDU Release 2 and 3 to evolve the platform, and **Domain APIs** are arriving to encourage Application development on OSDU.

DOMAIN APIs ARE THE KEY TO EFFICIENT WORKFLOWS IN APPLICATIONS

Domain APIs provide an Exploration and Production (E&P) domain specific way of accessing the system. The domain API signature carries semantics not in a generic API, making them a powerful tool for application developers who want the very best performance for accessing data. We may need to update this type-safe optimized API every few years, but they allow storage optimization based on access patterns over generic APIs.

What does this mean for generic APIs? Generic APIs never go away. They provide data neutral support for accessing metadata and content that are independent of the underlying structure. They can be used whether you want a log, well, seismic file, or document. They tend to be used by people focused on data management or simply data navigation (following references and parsing metadata) because these APIs have no opinion on the structure, content, or organization of data. For these data oriented users, when we add new data types to the platform through an extension, we don't need to change any code. However, generic APIs are not optimal for applications.

With a **Domain API,** I can get optimized access to data the reflects the problem that I am trying to solve. However, this imposed structure has to be understood and reflected in the Application. And this is desirable because that structure reflects the problem the application is trying to solve and thus is more intuitive.