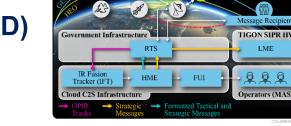
# **Rapid Application Deployment (RAD) Enterprise Service Bus**



SE DATA SHEET

# Rapid Application Deployment (RAD) Framework

INTRODUCTION: The RAD framework is a collection of open source tools that support the development, evaluation, testing, deployment, and operation of individual geospatial processing applications. RAD provides the Enterprise Service Bus (ESB) enabling high-volume, lowlatency application messaging.

# **Features & Benefits**

- Layered, platform-based approach abstracts the use of any specific vendor
- Free and Open-Source Software approach to answer mission needs
- Infrastructure neutral and can be deployed as individual services on AWS/C2S cloud infrastructure, virtual machines, or dedicated operational hardware.
- Represents maturation of common community approaches with optimizations based on operational lessons learned

Enterprise Service Bus	Multi-terabyte event driven messaging system running on opensource and IC proven architecture.	Kafka.
Build Automation	Automated, repeatable Continuous Integration/Continuous Development. Orchestrating and promoting versions of software through quality gates into production	Jenkins iii Nexus
Deployment Automation	Configuration management and DevSecOps orchestration. Infrastructure- as-code makes it highly repeatable and quickly replicated across domains.	SALTSTACK ANSIBLE
Test Automation	Automate as much of the test pipeline as. Scenarios written during design and continuously, automatically executed against each stage of development and version.	JMeter Opress
Security Automation	Automated vulnerability, coding anti-patterns and bug scanning included to remedy potential security vulnerabilities prior to code deployment.	Sonarqube
Platform Automation	Microservices and containerization enable a language and infrastructure agnostic container orchestration platform to dynamically auto-scale and manage services.	Cocker S FABRIC Kubernetes
Environment Automation	Automation of environment setup via Infrastructure as Code ensures no interruption to operations as new capabilities are developed, tested, and seamlessly fielded	ANSIBLE OpenCL Stamazon Webservices*

# RAD

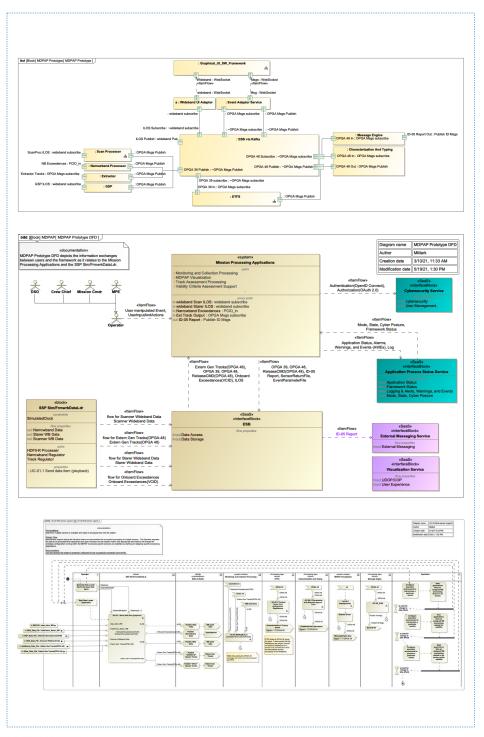
- Microservices
   Enables dynamic scaling of individual services that do one thing and do it well. These services can be configured, reconfigured, and reused based on mission needs
- Containers (ex. Docker)
   Enables automation and allows a service to be deployed as a single package that can run on a developer's laptop, AWS, C2S or bare metal.
- Platform as a Service (PaaS) Performs dynamic scaling of the services, to enable high performance, availability and reliability
- DevOps Continuous
   Integration/Continuous Deployment
   Enabling automated, rapid build and delivery of services to the operational environment with a fully automated
   CI/CD pipeline with integrated automated testing

**Open Interfaces** Leveraging industry open standard messaging technologies and standards based message definitions enable plug and play application integration.



# **Enterprise Service Bus (ESB)**

Our layered approach to the framework provides abstraction at all levels. The use of containerization allows for any service or tool to be refactored as an independent, 12-factor app, containerized and deployed on the framework. Many open source and commercial tools are already available as containers and can be added with minimal changes. Maxar's Enterprise Service Bus enables decoupled communication between the application services (containers or pods). The ESB uses commercially proven, industry standard open-source Kafka. Kafka is a distributed publish-subscribe messaging system that was created as a fast, scalable, and durable alternative to existing solutions. It is designed to broker enormous message streams for extremely low-latency analysis within cloud platforms. Kafka allows the ESB to be extremely robust, while providing a simple API and high throughput. The ESB leverages existing community-wide messaging standards (such as OPGA) to maximize compatibility with external systems and legacy applications.



MXR-DS-NAME 05/20

### ESB

#### • Scalable

Seamless scaling with the ability to handle many terabytes of data ensures a truly scalable solution.

#### Reliable

Supports multiple users while also replicating data as needed. The ESB automatically balances consumers in the event of failure, which other event driven services do not.

## Durable

The ESB provides intra-cluster replication by keeping messages on the disk to ensure reliability.

## Upgradeable

By decoupling the services from the bus independently the ESB is able to upgrade each service individually without causing the whole system to have downtime.

### Performant

Capable of handling multiple terabytes of data with rapid thruput for both consumers and producers. This is accomplished using disk structures that can offer constant high levels of performance.

# • Flexible

The ESB's use of containerization and micro-service design means that new services can be individually developed, elevated, and deployed without vendorspecific software and minimizes impact in the case of a change to a new hardware or software provider.

