BluSoftwave: Force Instrumenter

Package Design

Bill Anderson |CTO
bill.anderson@blusoftwave.com

Table of Contents

What is Force Instrumenter?

Force Instrumenter was designed to allow Realtime Observability of your Salesforce Transactions (Apex or Flow). There was no framework or tool that provided a means to allow Realtime inspection of transactions. Either one would eventually get the issues in an email, 24 hours later or not at all.

There was also a lack of discipline in development to Shift Left. Shift left is the practice of moving testing, quality, and performance evaluation early in the development process. Force Instrumenter allows a developer the ability to piece together a transaction with Tracers. These Tracers support Command, Query and Service entities. This helps understand the basic flow, and baseline resources and metrics without performing any heavy development.

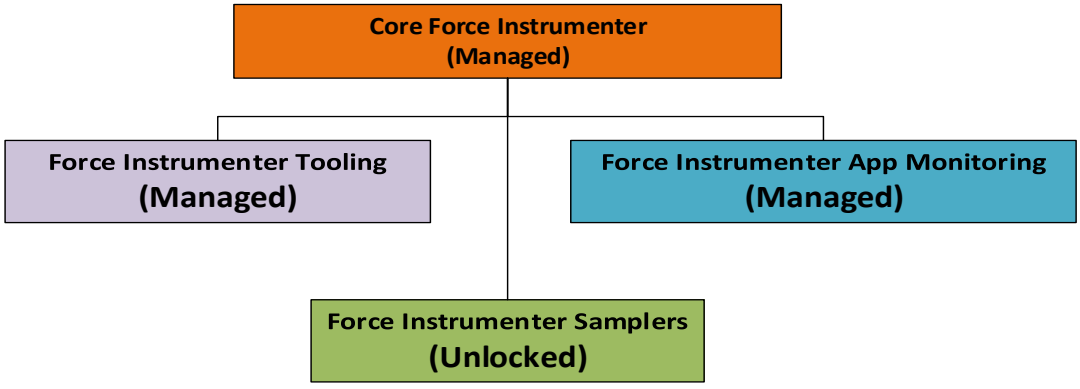
At BluSoftwave, we pride ourselves on delivering measurable quality with a proven architectural design. This document walks through the Package Design.

Overview

Force Instrumenter is built on three packages. Each package source is sub-divided to support various concerns (logging, configuration, common interfaces, repositories, etc.). The division in the packages source tree provides several advantages. Especially, when it comes to deployments and updates as this provides a focus aspect.

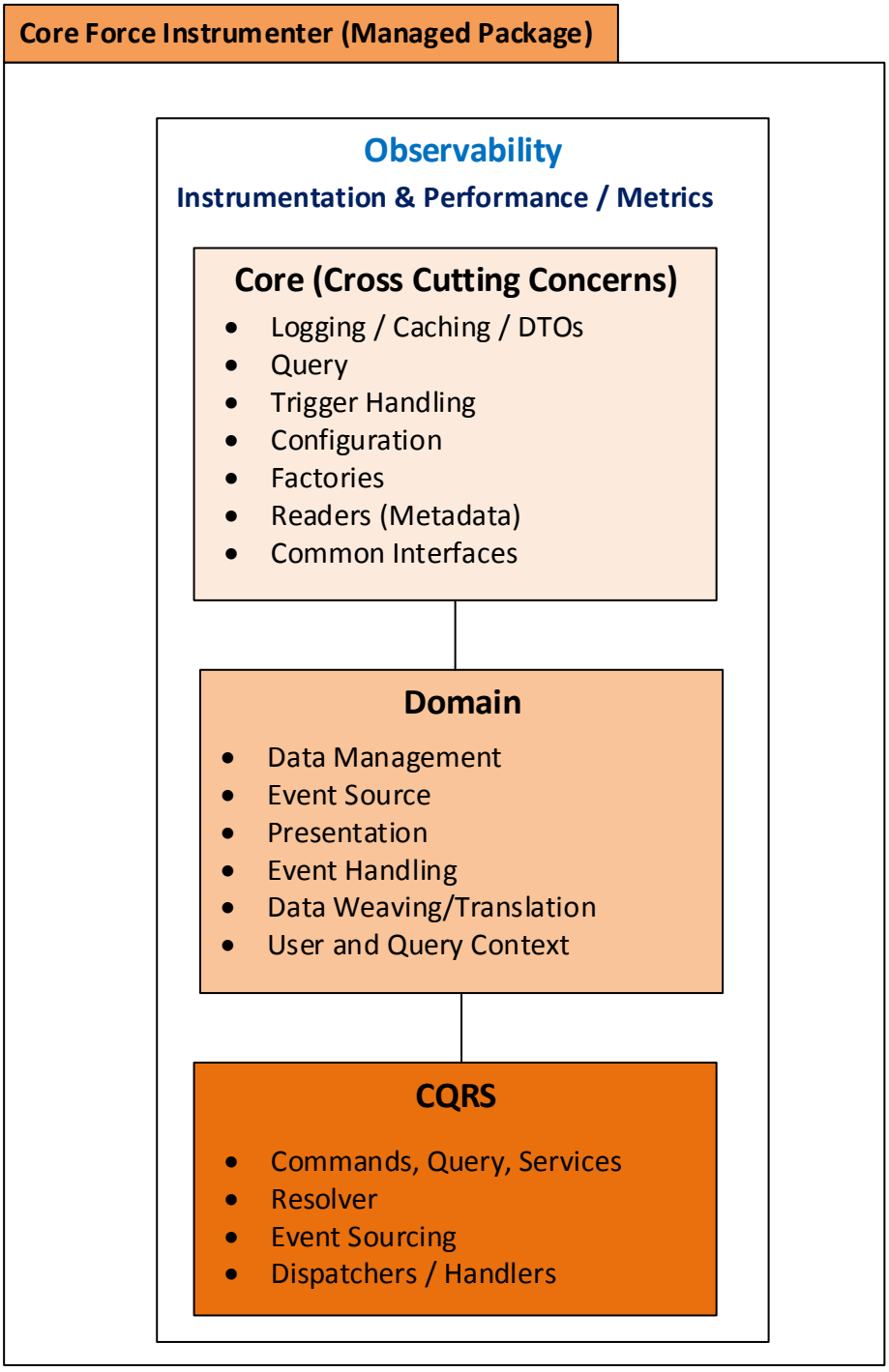
The separation of concerns allows the ability to plug-n-play. Layers follow a strict-layered architecture with interfaces guarding the boundaries. This allows for implementation to vary while preserving the design contract. In addition, packaging allows targeting specific Salesforce Orgs.

The three packages and one Samples package provides the current offering of ***Force Instrumenter***. The Samples package is not a core component of ***Force Instrumenter.***



Force Instrumenter: Core

The Core of ***Force Instrumenter*** contains the overall Observability and Instrumentation. The package source tree is further segregated into Core, Domain and CQRS. Each component supports common functionality utilized in other packages.

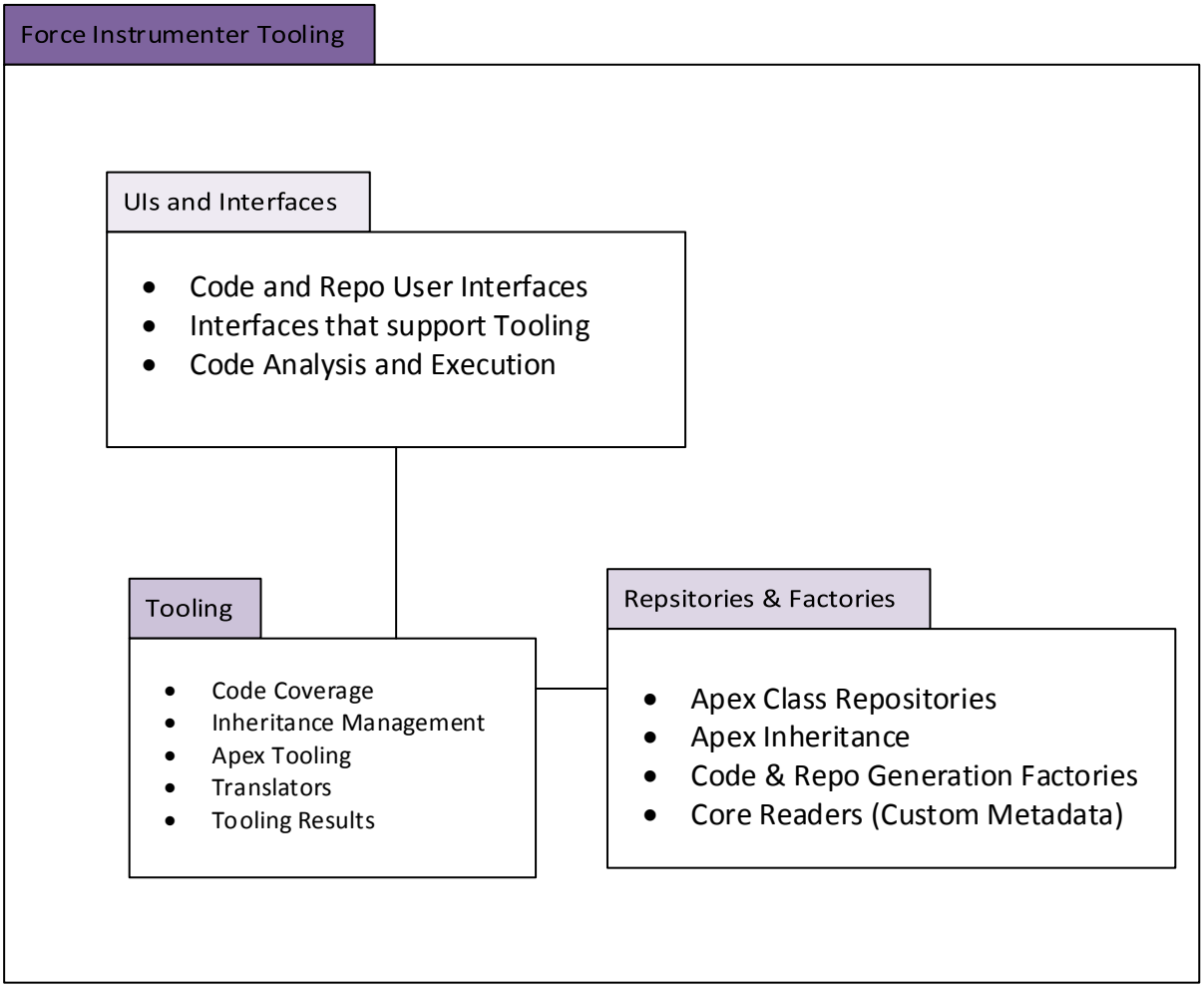


Force Instrumenter: Tooling

The Tooling of Force Instrumenter provides the ability to analyze, construct, run and test. As was done with other packages, this source tree if further segregated into User Interfaces, Common Interfaces, tooling, repositories and factories.

Tooling allows for a plethora of functionalities,

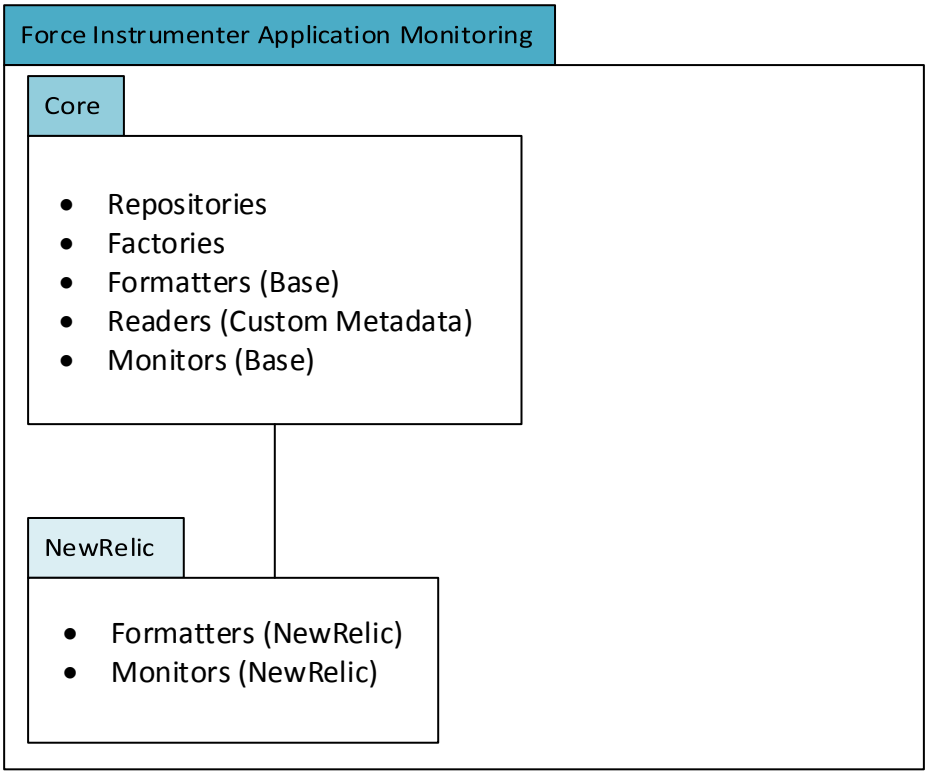
- Code generation such as specific Repositories, Commands, Queries and Services (CQS) along with their Unit Tests.
- Running of Commands, Queries, Services and ApexRunner[1]
- Create new Services from CQS files
- Create Sequence Diagrams of CQS files
- Provide specific Charting of Code under Observation



[1] Apex Runners provide the ability to run various components for a prolonged period to ascertain performance

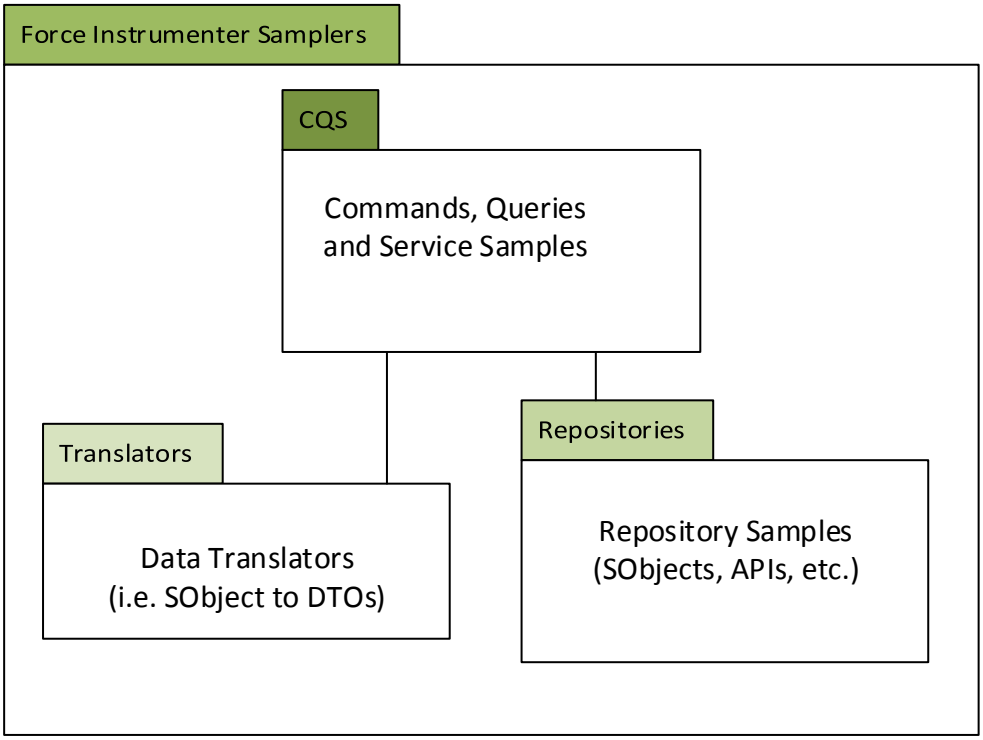
Force Instrumenter: Application Monitoring

The Application Monitoring of Force Instrumenter provides the ability to push data to various application monitoring's systems such as NewRelic, DataDog, etc. As was done with the other packages, this source tree is further segregated into Common Interfaces, formatters, readers and monitors. Each new monitor, i.e. NewRelic, is sub-directory utilizing the Core.



Force Instrumenter: Samples

The Samples of Force Instrumenter provides as the name states, samples. There is a common pattern followed in all CQS files. The examples show various Repositories for SObject and APIs. Also, how one can translate for one form to another (via Message Translators).



Summary

Force Instrumenter is meant to augment Observability on the left-side by placing the components in the hands of the Developers and/or Business Analysts (albeit, Apex, Flow) at the initial stages of Implementation and/or Proof of Concept. With packaging we provide the ability to address specific functionality without compromising your Software Development Lifecycle.

Packaging helps in this endeavor as you target their Salesforce Org appropriately. For example, you would NEVER place the Tooling Package into a Production Org. The table below provides guidance.

Package	Salesforce Org	Comments
Core	All Salesforce Orgs	Provides Observability
Tooling	Scratch | Sandboxes	Allows analyzes and code generation
Samples	Scratch | Sandboxes	Provides common patterns and practices
Application Monitoring	All Salesforce Orgs	Provides ability to push data to common Application Monitors