



Open Source Enterprise Integration Platform

Jeff Mitchell & Jun Suetake
ikasan.org
January 2017

- The Problem Domain
- Architecture Strategies
- Translating the Architecture
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary

- **[Domain]** Enterprise Application Integration (EAI) space
 - Integration of business systems in a landscape of interwoven and often complex finance processes

- **[Issue]** EAI is complex, costly, & not the primary business concern
 - Greater adoption of “Best of Breed” specialist applications
 - Business applications distributed across disparate platforms
 - Business data distributed across isolated silos
 - Legacy data repositories
 - Data duplication and integrity issues
 - No clear business data owner
 - Exponential integration requirements
 - Greater complexity of business demand on data orchestration
 - EAI is more than simply connecting applications

- **[Goal]** To provide simple, robust, configurable commoditised solutions
 - Expose business artifacts whilst isolating the integration specifics

- **Problem Domain**
- Architecture Strategies
- Translating Architecture
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary

- **Simplicity**
 - Must not require in-depth proprietary knowledge
 - Avoidance of overly complex or heavyweight frameworks
 - Highly testable simple constructs
- **Commonality**
 - Reusable interchangeable constructs
 - Standard contracts of interaction
 - Simple repeatable implementation steps
- **Adaptability**
 - Ability to support any type of business entity
 - Ability to integrate any type of Enterprise Information System
 - Without breaking the first two strategies of Simplicity & Commonality
- **Robust & Guaranteed operation**
 - Maintain data integrity in business delivery and failure scenarios
 - Require minimal manual intervention i.e. failure / automated recovery

- Problem Domain
- **Architecture Strategies**
- Translating Architecture
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary

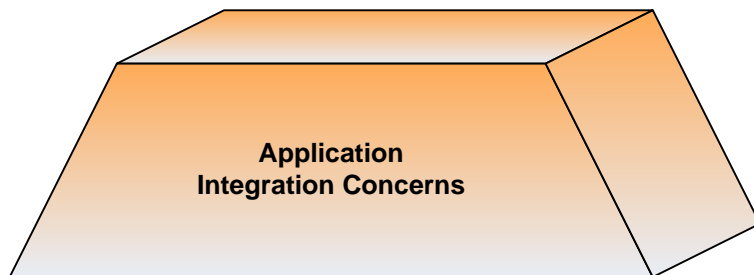
- Clear contract definition and separation of concerns

- Problem Domain
- **Architecture Strategies**
- Translating Architecture
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary

Architecture Strategies

- Clear contract definition and separation of concerns
 - Application concerns

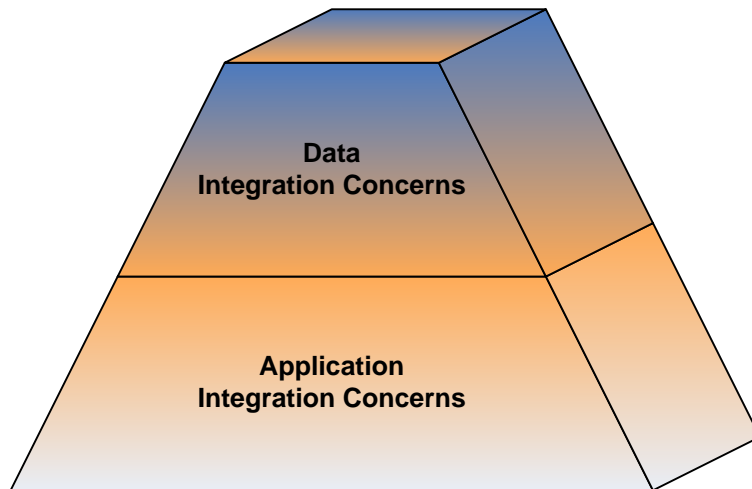
- Problem Domain
- **Architecture Strategies**
- Translating Architecture
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary



- **[Application]** performant, robust & guaranteed operation

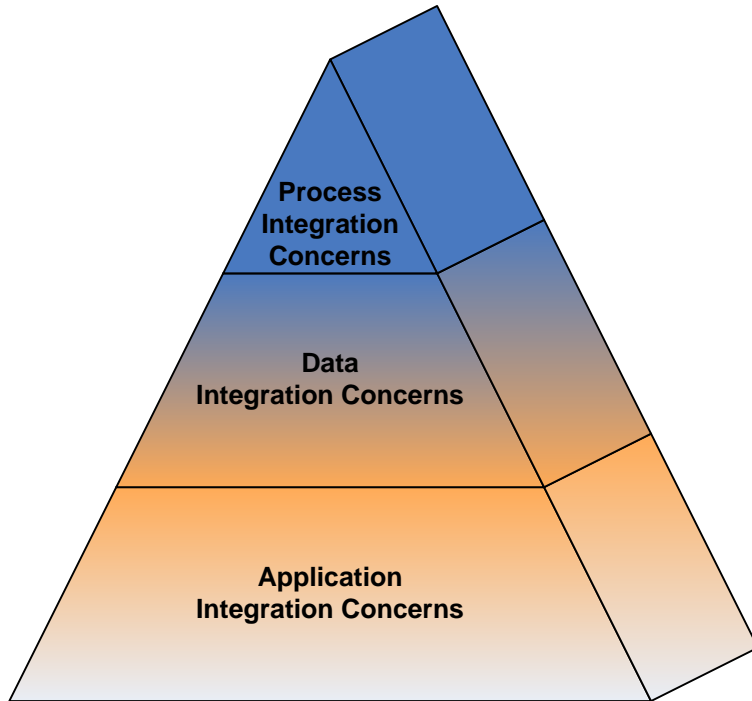
- Clear contract definition and separation of concerns
 - Application concerns
 - Data concerns

- Problem Domain
- **Architecture Strategies**
- Translating Architecture
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary



- **[Data]** presentation of standard, meaningful business data
- **[Application]** performant, robust & guaranteed operation

- Clear contract definition and separation of concerns
 - Application concerns
 - Data concerns
 - Business process concerns



- **[Process]** business event & entity orchestration (STP is the goal)
- **[Data]** presentation of standard, meaningful business data
- **[Application]** performant, robust & guaranteed operation

- Problem Domain
- **Architecture Strategies**
- Translating Architecture
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary

- Loose coupling
 - Integrations should only require knowledge of each other through exchange of business artifacts
- Tight cohesion
 - An integration's specifics must not bleed out unnecessarily
- High visibility
 - Business data tracking
 - Fail scenarios
 - Operational audit
- Single logical point of integration
 - Geographically agnostic / cross platform support
- Open standards aligned with proven design patterns
 - Adoption of industry standard Enterprise Design Patterns
 - Avoidance of vendor lock-in

- Problem Domain
- **Architecture Strategies**
- Translating Architecture
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary

[Key Strategies]

▪ Single Point of Integration	▪ Loose Coupling	▪ Variety of Standard Contracts
▪ Separation of Concerns	▪ Tight Cohesion	▪ Standard Design Patterns
▪ Interchangeable Constructs	▪ Highly Testable Constructs	▪ Data Integrity
▪ Control & Management	▪ Monitor & Alerting	▪ Minimal Manual Intervention
▪ Support any Protocol/Entity	▪ Business Event Tracking	▪ Operational Audit

- Problem Domain
- **Architecture Strategies**
- Translating Architecture
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary

- Ensure traceability from architecture through to implementation
 - All too often architecture fails to translate into the real world
 - The right architecture can reduce complexity

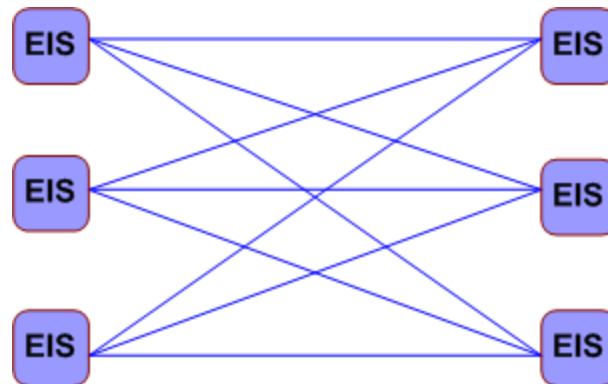
[Key Strategies]

▪ Single Point of Integration	▪ Loose Coupling	▪ Variety of Standard Contracts
▪ Separation of Concerns	▪ Tight Cohesion	▪ Standard Design Patterns
▪ Interchangeable Constructs	▪ Highly Testable Constructs	▪ Data Integrity
▪ Control & Management	▪ Monitor & Alerting	▪ Minimal Manual Intervention
▪ Support any Protocol/Entity	▪ Business Event Tracking	▪ Operational Audit

- Problem Domain
- Architecture Strategies
- **Translating Architecture**
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary

[Issue] Commonly agreed that “spaghetti integration” is bad

- Integration can be exponentially complex
- Bleed out of APIs, data syntax and business semantics
- Ripple effect of change is massive



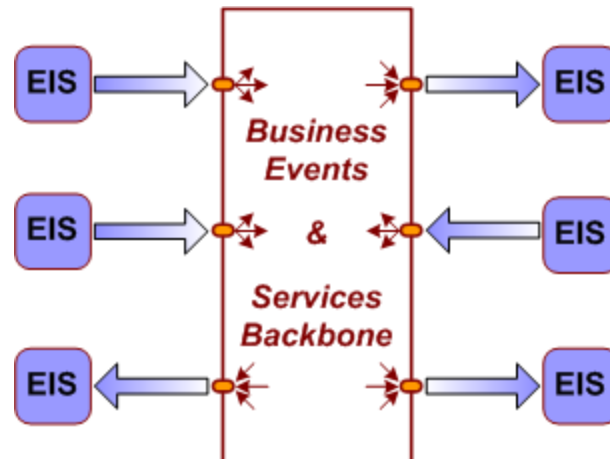
[Key Strategies]

▪ Single Point of Integration	▪ Loose Coupling	▪ Variety of Standard Contracts
▪ Separation of Concerns	▪ Tight Cohesion	▪ Standard Design Patterns
▪ Interchangeable Constructs	▪ Highly Testable Constructs	▪ Data Integrity
▪ Control & Management	▪ Monitor & Alerting	▪ Minimal Manual Intervention
▪ Support any Protocol/Entity	▪ Business Event Tracking	▪ Operational Audit

- Problem Domain
- Architecture Strategies
- **Translating Architecture**
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary

[Concept] Standard Enterprise Application Integration (EAI) approach

- Provision of an Event/Service backbone
- Single point of integration for EIS business flow
- Ripple effect of change is localised (*assuming best practice*)



[Key Strategies]

▪ Single Point of Integration	▪ Loose Coupling	▪ Variety of Standard Contracts
▪ Separation of Concerns	▪ Tight Cohesion	▪ Standard Design Patterns
▪ Interchangeable Constructs	▪ Highly Testable Constructs	▪ Data Integrity
▪ Control & Management	▪ Monitor & Alerting	▪ Minimal Manual Intervention
▪ Support any Protocol/Entity	▪ Business Event Tracking	▪ Operational Audit

- Problem Domain
- Architecture Strategies
- **Translating Architecture**
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary

[First Step] Modelling the Business Domain

- Defines fundamental entities, constructs, and their relationships
- Leverage existing domain knowledge
- External domain knowledge – we are not the first...
 - Enterprise Integration Design Patterns
 - [Ref. Enterprise Integration Patterns, Gregor Hohpe & Bobby Woolf, 2003]
 - Enterprise Application Design Patterns
 - [Ref. Patterns of Enterprise Application Architecture, Martin Fowler]
- Goal is a Rich Domain Model (data & behavior)
 - Anemic Domain Models (data only) fail over time

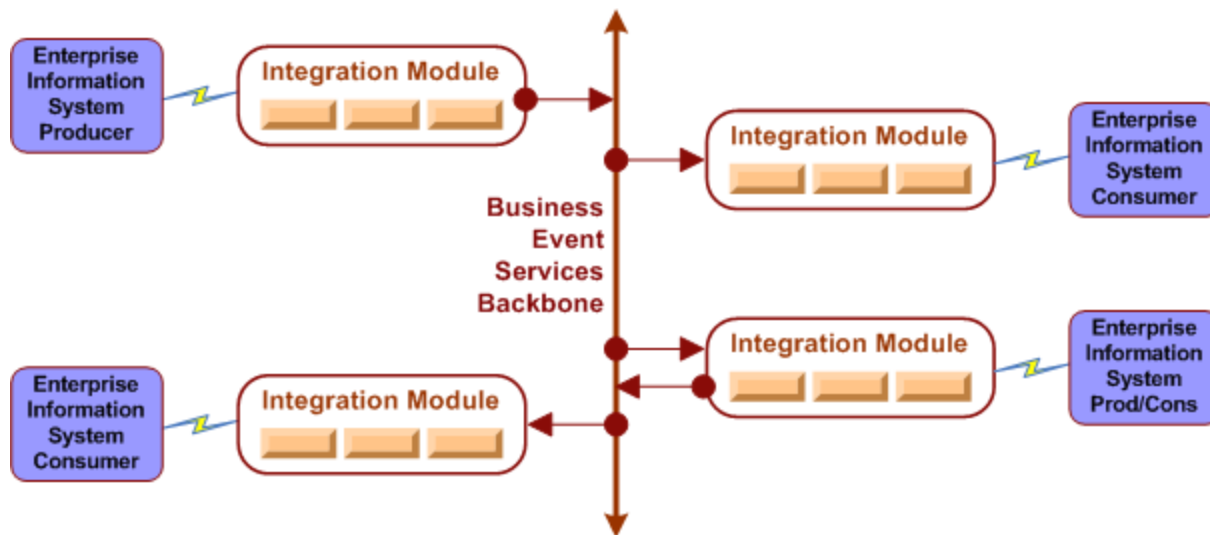
[Key Strategies]

✓ Single Point of Integration	✓ Loose Coupling	✓ Variety of Standard Contracts
<ul style="list-style-type: none"> ▪ Separation of Concerns 	<ul style="list-style-type: none"> ▪ Tight Cohesion 	<ul style="list-style-type: none"> ▪ Standard Design Patterns
<ul style="list-style-type: none"> ▪ Interchangeable Constructs 	<ul style="list-style-type: none"> ▪ Highly Testable Constructs 	<ul style="list-style-type: none"> ▪ Data Integrity
<ul style="list-style-type: none"> ▪ Control & Management 	<ul style="list-style-type: none"> ▪ Monitor & Alerting 	<ul style="list-style-type: none"> ▪ Minimal Manual Intervention
<ul style="list-style-type: none"> ▪ Support any Protocol/Entity 	<ul style="list-style-type: none"> ▪ Business Event Tracking 	<ul style="list-style-type: none"> ▪ Operational Audit

- Problem Domain
- Architecture Strategies
- **Translating Architecture**
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary

[Construct] Integration Modules

- Provides a logical grouping of business operations as a single integration point
- Provide either a source, target, or bi-directional business flow.



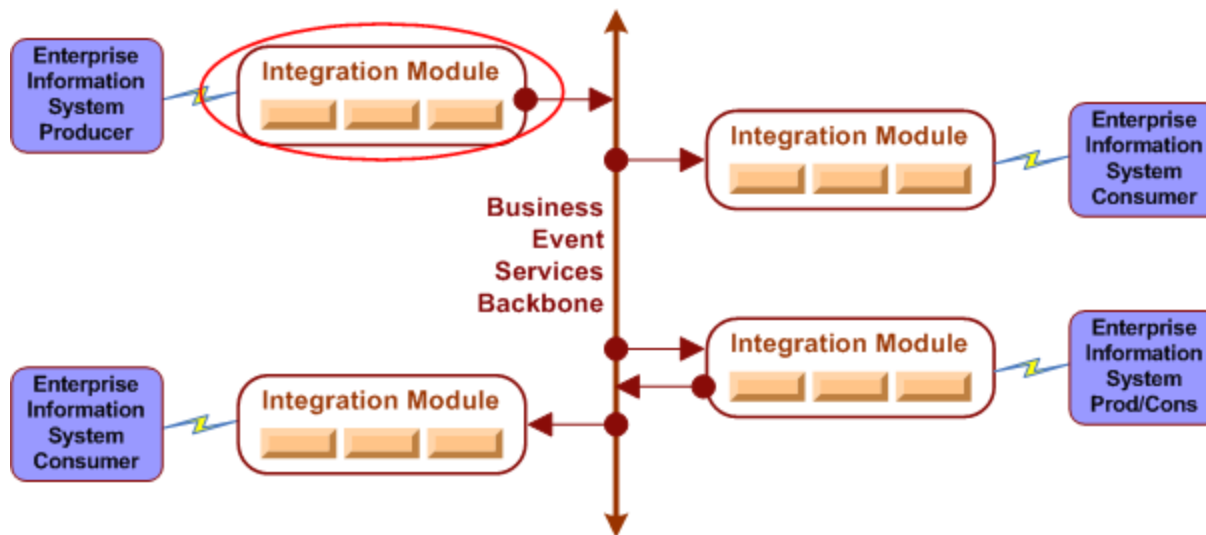
[Key Strategies]

✓ Single Point of Integration	✓ Loose Coupling	✓ Variety of Standard Contracts
<ul style="list-style-type: none"> ▪ Separation of Concerns 	<ul style="list-style-type: none"> ▪ Tight Cohesion 	<ul style="list-style-type: none"> ▪ Standard Design Patterns
<ul style="list-style-type: none"> ▪ Interchangeable Constructs 	<ul style="list-style-type: none"> ▪ Highly Testable Constructs 	<ul style="list-style-type: none"> ▪ Data Integrity
<ul style="list-style-type: none"> ▪ Control & Management 	<ul style="list-style-type: none"> ▪ Monitor & Alerting 	<ul style="list-style-type: none"> ▪ Minimal Manual Intervention
<ul style="list-style-type: none"> ▪ Support any Protocol/Entity 	<ul style="list-style-type: none"> ▪ Business Event Tracking 	<ul style="list-style-type: none"> ▪ Operational Audit

- Problem Domain
- Architecture Strategies
- **Translating Architecture**
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary

[Construct] Integration Modules

- Provides a logical grouping of business operations as a single integration point
- Provide either a source, target, or bi-directional business flow.



Translating the Architecture

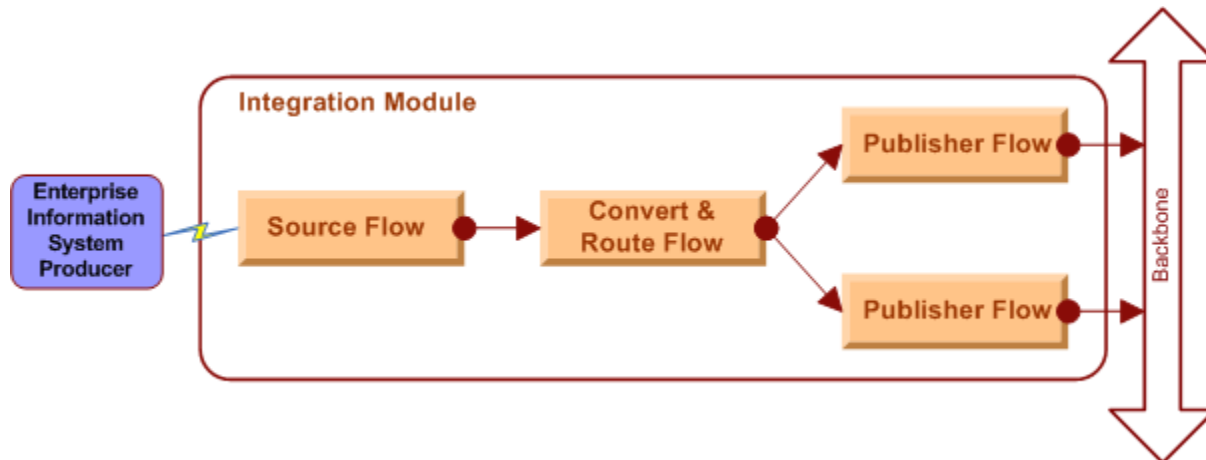
[Key Strategies]

✓ Single Point of Integration	✓ Loose Coupling	✓ Variety of Standard Contracts
✓ Separation of Concerns	✓ Tight Cohesion	✓ Standard Design Patterns
▪ Interchangeable Constructs	▪ Highly Testable Constructs	▪ Data Integrity
▪ Control & Management	▪ Monitor & Alerting	▪ Minimal Manual Intervention
▪ Support any Protocol/Entity	▪ Business Event Tracking	▪ Operational Audit

- Problem Domain
- Architecture Strategies
- **Translating Architecture**
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary

[Construct] Flows

- Provide cohesive operations on a business artifact as a synchronous operation
- Multiple flows can be chained to isolate concerns
- Standard event container allows any data type to be transported



Translating the Architecture

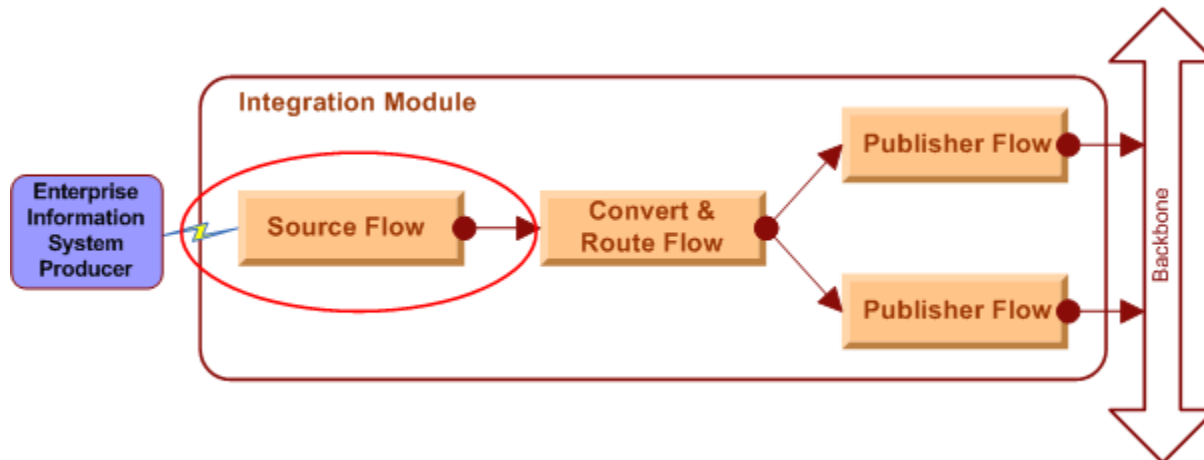
[Key Strategies]

✓ Single Point of Integration	✓ Loose Coupling	✓ Variety of Standard Contracts
✓ Separation of Concerns	✓ Tight Cohesion	✓ Standard Design Patterns
▪ Interchangeable Constructs	▪ Highly Testable Constructs	▪ Data Integrity
▪ Control & Management	▪ Monitor & Alerting	▪ Minimal Manual Intervention
▪ Support any Protocol/Entity	▪ Business Event Tracking	▪ Operational Audit

- Problem Domain
- Architecture Strategies
- **Translating Architecture**
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary

[Construct] Flows

- Provide cohesive operations on a business artifact as a synchronous operation
- Multiple flows can be chained to isolate concerns
- Standard event container allows any data type to be transported



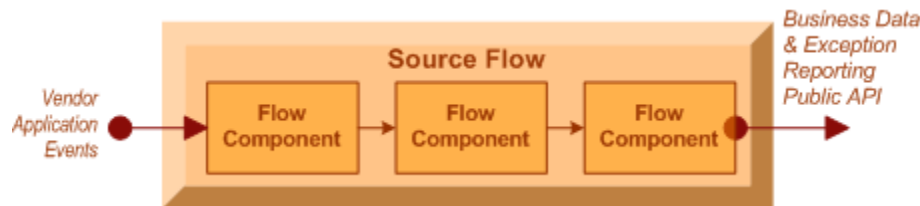
[Key Strategies]

✓ Single Point of Integration	✓ Loose Coupling	✓ Variety of Standard Contracts
✓ Separation of Concerns	✓ Tight Cohesion	✓ Standard Design Patterns
✓ Interchangeable Constructs	✓ Highly Testable Constructs	✓ Data Integrity
▪ Control & Management	▪ Monitor & Alerting	▪ Minimal Manual Intervention
▪ Support any Protocol/Entity	▪ Business Event Tracking	▪ Operational Audit

- Problem Domain
- Architecture Strategies
- **Translating Architecture**
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary

[Construct] Flow Components

- POJO implementations provide independent, interchangeable operations
- Isolated component white box testing and chained black box testing
- All flow components within an atomic operation guaranteeing data integrity
- Unfettered event transport without casting or serialization overheads



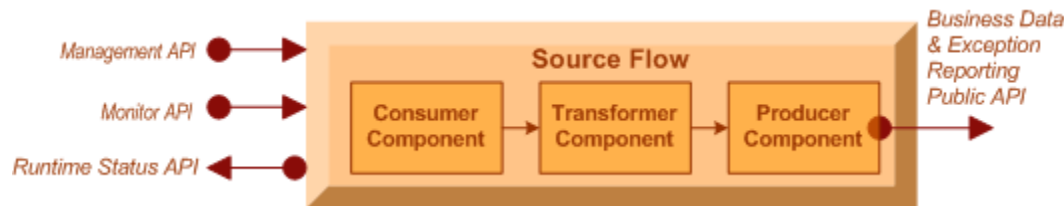
[Key Strategies]

✓ Single Point of Integration	✓ Loose Coupling	✓ Variety of Standard Contracts
✓ Separation of Concerns	✓ Tight Cohesion	✓ Standard Design Patterns
✓ Interchangeable Constructs	✓ Highly Testable Constructs	✓ Data Integrity
✓ Control & Management	✓ Monitor & Alerting	✓ Minimal Manual Intervention
▪ Support any Protocol/Entity	▪ Business Event Tracking	▪ Operational Audit

- Problem Domain
- Architecture Strategies
- **Translating Architecture**
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary

[Construct] Flow Contracts

- Management API provides business flow control (start, stop, pause, resume)
 - Pause/Resume for fine grained consumer component resource management only
 - Stop/Start for course grained general component resource management
- Runtime Status API provides real-time health status (running, stopped, recovery, stoppedInError)
- Monitor API provides pluggable runtime health notifiers i.e. (Email, Zenoss, Hobbit, SMS, etc)
- Recovery Manager for configurable exception handling for automated flow recovery
 - Invocable by any component to change flow state



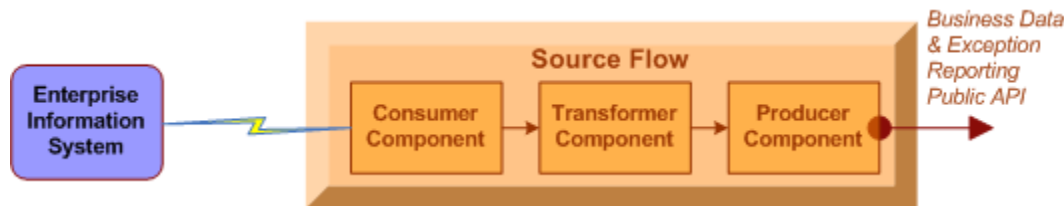
[Key Strategies]

✓ Single Point of Integration	✓ Loose Coupling	✓ Variety of Standard Contracts
✓ Separation of Concerns	✓ Tight Cohesion	✓ Standard Design Patterns
✓ Interchangeable Constructs	✓ Highly Testable Constructs	✓ Data Integrity
✓ Control & Management	✓ Monitor & Alerting	✓ Minimal Manual Intervention
✓ Support any Protocol/Entity	▪ Business Event Tracking	▪ Operational Audit

- Problem Domain
- Architecture Strategies
- **Translating Architecture**
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary

[Construct] EIS / Application Integration

- Consumers can support any type of protocol / API
- Extend guaranteed integrity to non-guaranteed EIS sources/targets
- No proprietary knowledge required; only EIS API
- Resource adapters pluggable re-usable components (Java Connector Architecture spec)
- 3rd party off-the-shelf or internally developed



[Key Strategies]

✓ Single Point of Integration	✓ Loose Coupling	✓ Variety of Standard Contracts
✓ Separation of Concerns	✓ Tight Cohesion	✓ Standard Design Patterns
✓ Interchangeable Constructs	✓ Highly Testable Constructs	✓ Data Integrity
✓ Control & Management	✓ Monitor & Alerting	✓ Minimal Manual Intervention
✓ Support any Protocol/Entity	✓ Business Event Tracking	✓ Operational Audit

- Problem Domain
- Architecture Strategies
- **Translating Architecture**
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary

[Construct] Administration Console

- Flow management
 - runtime status (stopped, running, recovery, stoppedInError)
 - runtime control (start, stop, pause, resume)
 - deployment control (start-up type {auto, manual, disabled})
 - flow component chain view and drill down
 - audit log of flow actions (stop/start/pause/resume)
- Wiretap management
 - dynamic creation/deletion of real-time in-flight data capture
- User management
 - creation, deletion, and permission maintenance

[Key Strategies]

✓ Single Point of Integration	✓ Loose Coupling	✓ Variety of Standard Contracts
✓ Separation of Concerns	✓ Tight Cohesion	✓ Standard Design Patterns
✓ Interchangeable Constructs	✓ Highly Testable Constructs	✓ Data Integrity
✓ Control & Management	✓ Monitor & Alerting	✓ Minimal Manual Intervention
✓ Support any Protocol/Entity	✓ Business Event Tracking	✓ Operational Audit

- Problem Domain
- Architecture Strategies
- **Translating Architecture**
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary

[Construct] User Console

- In-flight business event searches
 - by module(s)
 - by flow(s)
 - by business stream(s)
 - parameter filters
 - business content, date/time, names, etc
- Event “*Life Identifier*” support for full end to end business event tracking
 - flow component chain view and drill down
 - audit log of flow actions (stop/start/pause/resume)
- User management
 - creation, deletion, and permission maintenance

[Key Strategies]

✓ Single Point of Integration	✓ Loose Coupling	✓ Variety of Standard Contracts
✓ Separation of Concerns	✓ Tight Cohesion	✓ Standard Design Patterns
✓ Interchangeable Constructs	✓ Highly Testable Constructs	✓ Data Integrity
✓ Control & Management	✓ Monitor & Alerting	✓ Minimal Manual Intervention
✓ Support any Protocol/Entity	✓ Business Event Tracking	✓ Operational Audit

- Problem Domain
- Architecture Strategies
- **Translating Architecture**
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary

[Construct] Monitors & Notifiers

- Big Brother / Hobbit client
- Zenoss
- Email client
- Log scraper
- HTTP client

[Key Strategies]

✓ Single Point of Integration	✓ Loose Coupling	✓ Variety of Standard Contracts
✓ Separation of Concerns	✓ Tight Cohesion	✓ Standard Design Patterns
✓ Interchangeable Constructs	✓ Highly Testable Constructs	✓ Data Integrity
✓ Control & Management	✓ Monitor & Alerting	✓ Minimal Manual Intervention
✓ Support any Protocol/Entity	✓ Business Event Tracking	✓ Operational Audit

- Problem Domain
- Architecture Strategies
- **Translating Architecture**
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary

[Construct] OSS Test Frameworks & Ikesan FlowTestHarness

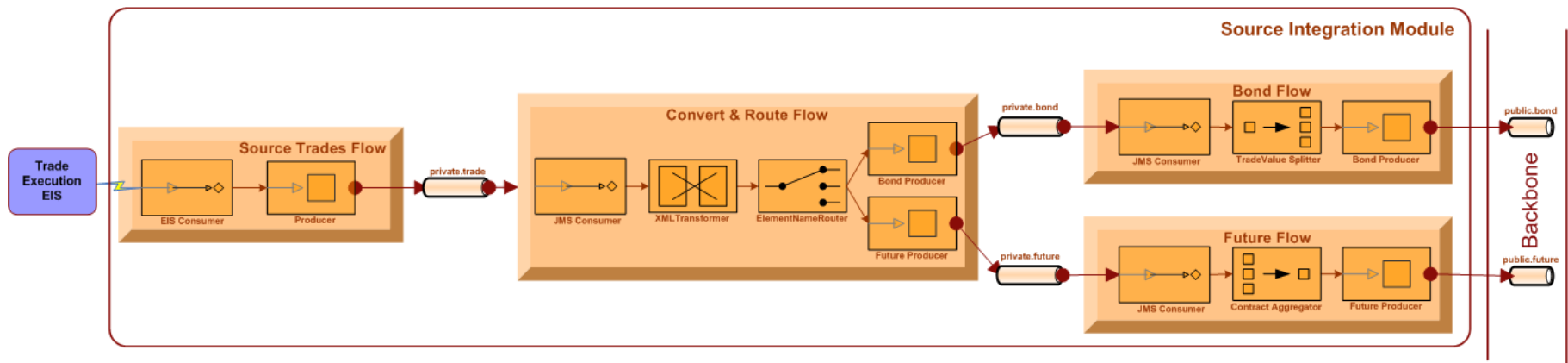
- Flow component automated unit testing
 - Junit, JMock, XMLUnit
- Initiator automated unit testing
 - JUnit, JMock
- Flow automated unit (integration) testing
 - Spring-test, Ikesan FlowTestHarness, HSQL & H2 in memory database
- Business Stream end to end testing
 - FitNesse

Translating the Architecture

■ Sample Source Integration Module

- Source Trades Flow
 - Get trades (CSV format) from EIS with minimal failure potential
- Convert & Route Flow
 - Transform CSV -> XML syntax; Transform XML semantics; Route on type
- Bond Flow
 - Split one bond event into many bond events based on trade size rule
- Future Flow
 - Aggregate multiple future contract events until “complete” then push as single event

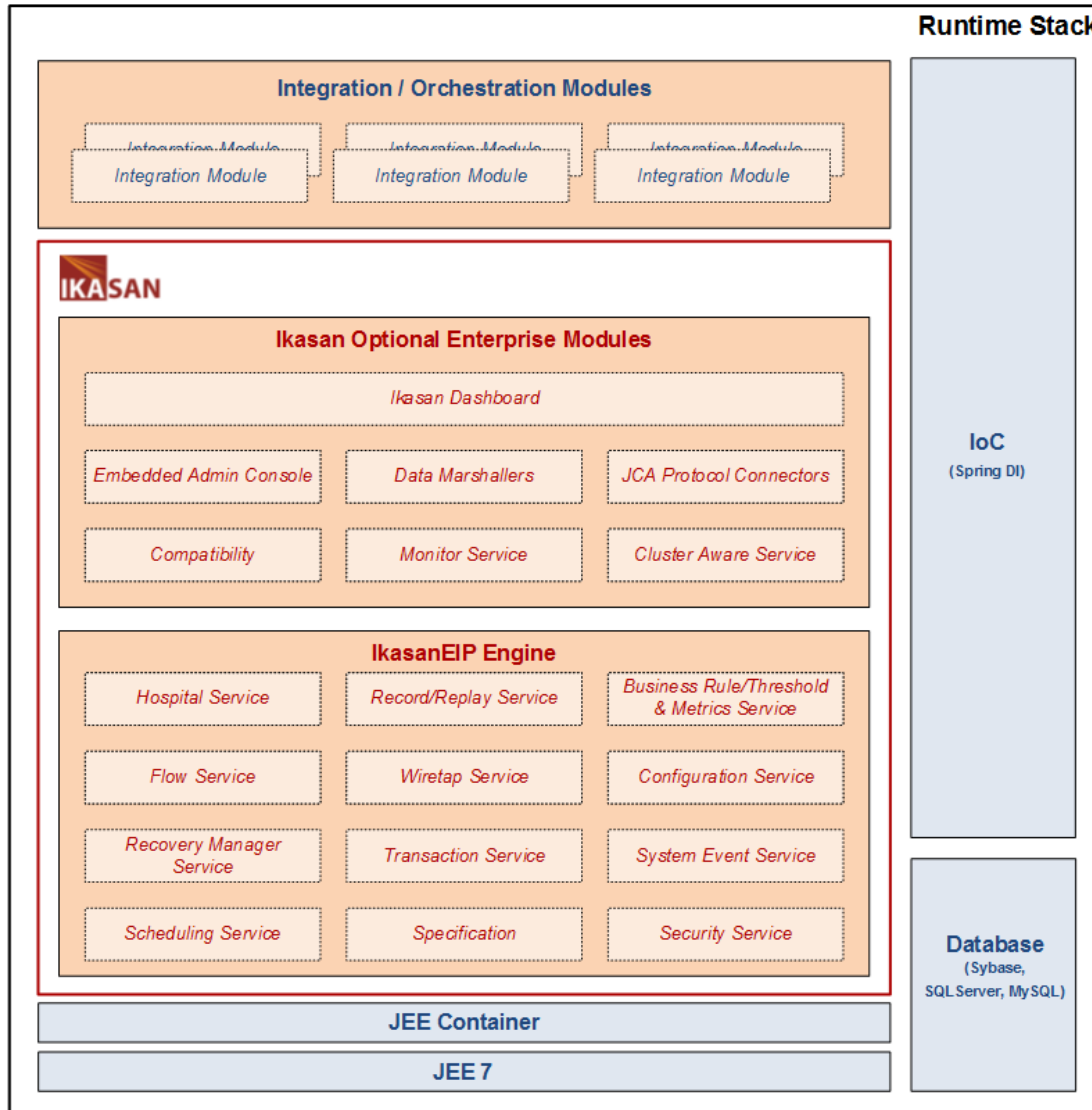
- Problem Domain
- Architecture Strategies
- **Translating Architecture**
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- Summary



Technology Stack



■ Runtime Stack



- Problem Domain
- Architecture Strategies
- Translating Architecture
- **Technology Stack**
- Features
- Development Lifecycle
- Roadmap
- Summary

■ Development Stack

■ Core technologies

- Java 1.7+ / JEE
- XML
- Spring 3.1.1.RELEASE
- Hibernate

■ Test Frameworks

- JUnit
- Jmock
- FitNesse
- SpringTest
- Ikesan Flow Test Harness

■ Scaffolding

- Source Management – GitHub
- Requirements and Issue Management – JIRA
- Peer Review and Acceptance Management – Crucible/FishEye
- Project Build Management – Maven
- Project Artefact Management – Sonotype Nexus
- Continuous Integration – Jenkins/Bamboo

- Problem Domain
- Architecture Strategies
- Translating Architecture
- **Technology Stack**
- Features
- Development Lifecycle
- Roadmap
- Summary

■ IkasanEIP Stack

■ IkasanEIP Engine

- ***IkasanEIP*** - engine underpinning all operations and services

■ Ikasan Optional Enterprise Modules

- ***Compatibility*** – provides runtime inter-op between Ikasan versions
- ***Monitor Service*** – default monitor & notifiers
- ***Off-the-Shelf Flow Components*** i.e. JSON <-> XML, XSLT, MongoDB, JMS, Quartz
- ***Clustering Service*** – makes Ikasan cluster aware
- ***Threshold Monitor & Alerts*** – business throughput monitoring
- ***Replay Service***
 - event record & replay
 - Module revert to point in time state & replay (TBC)
- ***Hospital Service***
 - Management of failed events for review/resolution resubmission/discard
 - Management of failed events & dependents
- ***Business Stream Templates***
 - Boilerplate commoditisation of standard business orchestration (TBC)

■ Ikasan Dashboard

- Problem Domain
- Architecture Strategies
- Translating Architecture
- **Technology Stack**
- Features
- Development Lifecycle
- Roadmap
- Summary

■ IkananEIP Stack

- IkananEIP is agnostic to the Operating System i.e. Linux, Windows, OSX
- IkananEIP runs In Container
 - Can run in any JEE compliant container
 - Proven in WebLogic 10+, JBoss 6+
- IkananEIP runs Standalone (outside container)
 - Can run standalone with reduced Enterprise Services
- IkananEIP Persistence
 - Proven with JDBC implementations ie. Sybase, SQLServer, MySQL, HSQLDB, H2
- Version 1.3.x current production release
- Stable production backbone since 2007

- Problem Domain
- Architecture Strategies
- Translating Architecture
- **Technology Stack**
- Features
- Development Lifecycle
- Roadmap
- Summary

■ Technical Integration

■ Off-the-Shelf Protocols

- JMS (HornetQ, JBossA-MQ, ActiveMQ, WebLogic, IBM WebSphere MQ)
- FTP (transactional & chunking support)
- SFTP (transactional & chunking support)
- RDBMS (via Hibernate ORM, Sybase, MS SQL, MySQL, Oracle)
- MQ Client
- FIX
- SMTP (Email)
- HTTP(s)

■ Data Constructs

- Event – Ikesan internal entity container
- EventIdentifier – Ikesan internal Event Life Identifier
- Timestamp – Ikesan internal Event creation timestamp
- Payload – actual entity in-flight – can be anything

- Problem Domain
- Architecture Strategies
- Translating Architecture
- Technology Stack
- **Features**
- Development Lifecycle
- Roadmap
- Summary

■ Functionality

- Flow Runtime Management – control at the runtime thread level
 - Allows a client to stop, start, pause, resume, enable, disable, query runtime status
- Flow Drivers
 - Real-time Event Driven
 - Batch Event Driven
 - Schedule Driven
- Flow Non-Functional Features
 - Full Transactional Semantics
 - Non-transactional
 - Local transactions
 - 2-Phase transactions
 - Last resource commit optimisation (LRCO)
 - incorporate non-transactional resources within a distributed transaction
 - Configurable Automated Recovery
 - Rollback operations & stop
 - Rollback operations and retry in x seconds
 - Rollback operations and retry a number of times or indefinitely
 - Report the exception and continue

- Problem Domain
- Architecture Strategies
- Translating Architecture
- Technology Stack
- **Features**
- Development Lifecycle
- Roadmap
- Summary

■ Functionality

■ Flow Components

■ Transformers

- XSLT
- XML Reader, Validator
- any POJO implementing Translator/Converter interface

■ Routers

- Recipient list
- Content
- Xpath
- Any POJO implementing Ikesan Router interface

■ Sequencers

- Aggregator
- Any POJO implementing Ikesan Sequencer interface

■ Splitters

- Any POJO implementing Ikesan Splitter interface

■ Filters

- Any POJO implementing Ikesan FilterRule/Filter interface

- Problem Domain
- Architecture Strategies
- Translating Architecture
- Technology Stack
- **Features**
- Development Lifecycle
- Roadmap
- Summary

■ Functionality

■ Flow Components

■ Endpoints

■ Consumers

- JMS Standalone
- JMS Container Based
- Scheduled
- Any resource adapter client or third-party API
- Any POJO implementing Ikasan Consumer interface

■ Brokers

- MongoDB
- JDBC
- Any resource adapter client or third-party API
- Any POJO implementing Ikasan Broker interface

■ Producers

- JMS Standalone
- JMS Container Based
- Loggers
- Any resource adapter client or third-party API
- Any POJO implementing Ikasan Producer interface

- Dynamically capture runtime data at any point in the flow
- Users able to search business content over Flows/Business Streams

- Problem Domain
- Architecture Strategies
- Translating Architecture
- Technology Stack
- **Features**
- Development Lifecycle
- Roadmap
- Summary

- Business Integration
 - Post Trade Execution
 - Bloomberg
 - Trade Feed
 - Consolidated Message Feed
 - Derivative Settlement Feed
 - Daily Trade Reporting
 - PATS Post Trade Feed
 - Espeed Post Trade feed
 - Tradeweb Post Trade Feed (US Treasuries & JGBs)
 - ION Post Trade Feed
 - Fidessa ETP Execution Report and Allocation FIX messages
 - ANVIL Repo Trading
 - Positions
 - ION
 - Bloomberg Position Feed
 - Intellimatch
 - Price Marks
 - ION (Price Marks)
 - Bloomberg (Price UMTM)

- Problem Domain
- Architecture Strategies
- Translating Architecture
- Technology Stack
- **Features**
- Development Lifecycle
- Roadmap
- Summary

- Business Integration
 - Trade Matching, Reporting & Reconciliation
 - TRAX
 - Bloomberg Daily Trade Reporting
 - Sales Commission Reporting
 - FIOS
 - Reference Data
 - Bloomberg Data License Per Security; Data License Back Office
 - MACE Convertible Bonds
 - Swap Monitor Financial Calendar Holidays
 - Fidessa ETP MIS & FDA
 - GoldenSource
 - Market Data
 - Markit CDS Service
 - Markit iBoxx Benchmark Indices
 - Bank of America Merrill Lynch Indices
 - Xenomorph Timescape

- Problem Domain
- Architecture Strategies
- Translating Architecture
- Technology Stack
- **Features**
- Development Lifecycle
- Roadmap
- Summary

- Business Integration
 - Settlements & Operations
 - SWIFT Alliance
 - GlobeOp
 - Ratings
 - Moodys
 - Standard & Poors
 - Compliance
 - Thomson Reuters TransWatch Securities Data
 - Corporate Actions
 - FTI Corporate Action & Securities Universe

- Problem Domain
- Architecture Strategies
- Translating Architecture
- Technology Stack
- **Features**
- Development Lifecycle
- Roadmap
- Summary

- Approach based on TDD/BDD Agile dogma
 - Gather/refine user stories and create use cases
 - Sketch the stories into a stream/module/flow design
 - Create/refine the Flow Test Harness reflecting the design
 - Create modules/flows/components to meet test harness acceptance
 - Peer/pair walk through (10 minute daily stroll)
 - Design reviews
 - Code reviews
 - Continuous Integration (CI) test coverage increases
 - Deliver functionality to users for proofs
 - Repeat!
- Recent Example Implementation Times
 - 1 day – Completed module delivery for raw to raw message transport (RMS: Raw Message Service);
 - 1 - 10 days – Completed module delivery for transformed XML (CMS: Common Message Service); real-time feed and batch aggregation;
 - 20 days – Completed delivery of complex business work flow (BPAWS: Business Process Application Workflow Service). Orchestrating Fixed Income Bond Trading flows between two business entities.

- Problem Domain
- Architecture Strategies
- Translating Architecture
- Technology Stack
- Features
- **Development Lifecycle**
- Roadmap
- Summary

- Version 1.3.0 released on January 2017 (Latest and Stable)
- Version 1.3.x (Bug Fixes for 1.3)
- Version 2.0.0 GA due June 2017
 - JDK 1.8+ Support
 - Integrate Apache Solr – blazing fast enterprise search platform
 - GUI view of Flows within end to end Business Streams
 - Micro services deployment topology (no Application Server required)

- Problem Domain
- Architecture Strategies
- Translating Architecture
- Technology Stack
- Features
- Development Lifecycle
- **Roadmap**
- Summary

■ Ikasan Project Resource Summary

- Web Site: <http://www.ikasan.org>
- GitHub: <https://github.com/ikasanEIP/ikasan>
- JIRA/ConfluenceWIKI/Bamboo: <http://ikasan.atlassian.net>
- Mailing List: <http://lists.sourceforge.net/lists/listinfo/ikasaneip-user>
- IRC: <https://ikasan.atlassian.net/wiki/display/IKASAN/IRC>

- Problem Domain
- Architecture Strategies
- Translating Architecture
- Technology Stack
- Features
- Development Lifecycle
- Roadmap
- **Summary**



Open Source Enterprise Integration Platform

Jeff Mitchell & Jun Suetake
ikasan.org
January 2017