



# > SOA Jumpstart for Federal Projects

IONA Technologies    August 2007

Jumpstart Guide



Making Software Work Together™

# Summary

Open source software removes the barrier of upfront investment, reduces the risk of vendor lock-in and provides greater flexibility for Federal agencies as they adopt and implement a service-oriented architecture (SOA). IONA's approach to SOA combines the speed and innovation of open source software with the reliability and expertise of commercially provided enterprise services.

This demonstration implements a complete SOA sandbox with a contrived Command and Control Center use case. The logic of the service implementations are very basic and are meant to illustrate general feasibility and serve as a hypothetical framework that can be used as a starting point to develop a SOA. Services can be added to this sandbox, or the existing services can be modified. The demonstration uses the VMWare Player as the delivery mechanism, avoiding the need to install and configure numerous software packages.

This Jumpstart package consists of three components: this Jumpstart Guide, a non-intrusive VMware Federal demonstration (downloadable or DVD), support and guidance from your local IONA sales team. Underlying the demonstration is IONA's family of distributed, open source SOA infrastructure products. It is expected that you will need less than three hours to read this document, install and run the demonstration.

[Federal Use Case Scenario](#)

[IONA Open Source Products](#)

[Installing the Jumpstart Appliance](#)

[Running the Jumpstart Appliance](#)

[Extending the Jumpstart Framework](#)

[Removing the Jumpstart Appliance](#)



## Federal Use Case Scenario

To better diagnose and respond to threats, several government systems, Web services, and mobile field assets have been integrated with a Command Center (CC) portal using open source middleware. This demonstration simulates the following systems:

- **Mobile Field Assets** – Tanks, platoons and helicopters deployed in the field.
- **Theater Operator** – military asset information and location and operator initiated threat alerts, e.g. this could be a border patrol sensor that registers a possible threat.

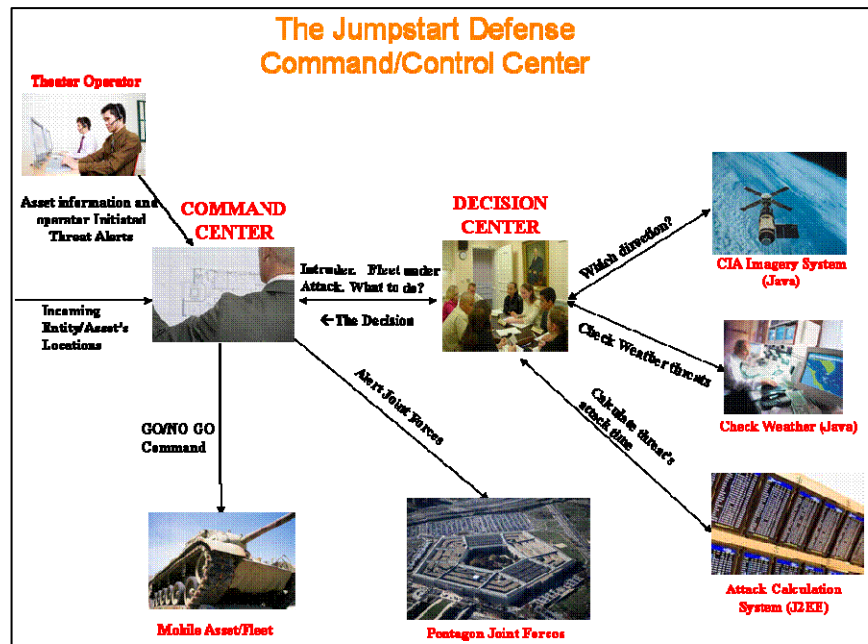
The following services are implemented as part of the architecture:

- **Geospatial imaging service** - surveys terrain to assess location and movement of military assets and threats, meant to represent the national imaging agency (fictitious names used)
- **Weather service** – understand weather patterns, meant to represent the weather agency (fictitious name used)
- **Attack Calculation Service** – estimates threat attack time, meant to represent complex algorithms
- **Decision Center** – correlate events from all systems, providing decisions
- **Google Map service** – provides map based on input coordinates to theater operator
- **Pentagon Notification service** – alerts Pentagon and other subscribing agencies of pending attacks

The CC displays a Google Map of mobile military assets and threats, refreshing every five seconds (configurable). A message is received that there is an unrecognized entity (badguy), the CC operator sends a threat alert to the Decision Center for further analysis. The Decision Center communicates with several services (Geospatial imaging, Weather, Attack Calculation Services) to see what the threat is, size of the threat, what direction the threat is moving, and how fast it is moving. If it is diagnosed as a threat, the decision needs to be made to have the nearest field asset flee or stay and endure possible pending battle. This decision is made by the decision center and sent back to the theater operator and the Pentagon if decision is “pending battle”.

The following diagram highlights the services and entities.





## IONA Open Source Products Used

IONA makes it easier to use open source SOA infrastructure by providing leading open source technologies in certified releases that are fully tested, validated and supported for enterprise use. IONA validates the interoperability of the products to give customers the most flexible approach to service-oriented integration.

IONA's family of distributed, open source SOA infrastructure products is built on four key products: the FUSE Message Broker, the FUSE ESB, the FUSE Services Framework, and the FUSE Mediation Router.

**This version of Jumpstart utilizes the FUSE Services Framework and future versions will use additional components.**



Making Software Work Together™

## FUSE Message Broker

The FUSE Message Broker is a high performance solution for reliable messaging based on the Apache ActiveMQ project, the leading open source JMS platform for enterprise messaging. The FUSE Message Broker provides high performance, unlimited scalability, and mission-critical reliability for distributed enterprise computing. As an open source solution, the FUSE Message Broker provides the foundation for a truly cost-effective and flexible messaging platform that can scale across the enterprise, reliably executing transactions and moving data, efficiently scaling operations, and connecting processes across heterogeneous database and application environments.

With connectivity via the STOMP and OpenWire protocols, the FUSE Message Broker extends to numerous non-Java environments, including clients in C/C++, .NET, perl, Python and Ruby. The FUSE Message Broker provides numerous options for performance optimization and persistence including working with any JDBC-compliant database. A lightweight broker that can be deployed to virtually any Java environment and embedded with applications, the FUSE Message Broker supports an enormous range of configurations, network topologies and distributed application architectures.

## FUSE ESB

Based on the Apache ServiceMix project, the FUSE ESB is the integration solution that frees architects from the dependencies among integration technologies that have traditionally locked enterprises into proprietary stacks. One of the first shipping implementations of the Java Business Integration (JBI) specification, the FUSE ESB supports any JBI-compliant binding for connectivity, and enables any JBI-compliant engine to be integrated into the SOA backplane for message processing. Any standards-compliant technology – from any commercial software vendor, bespoke systems developer, or open source project – can leverage the FUSE ESB to deploy to an organization's infrastructure.

The FUSE ESB can be deployed to any Java environment, and can be embedded in applications and deployed to endpoints via the Spring Framework. It supports a wide range of protocols and transports, and is provided with a number of JBI components for transformations and routing.

## FUSE Services Framework

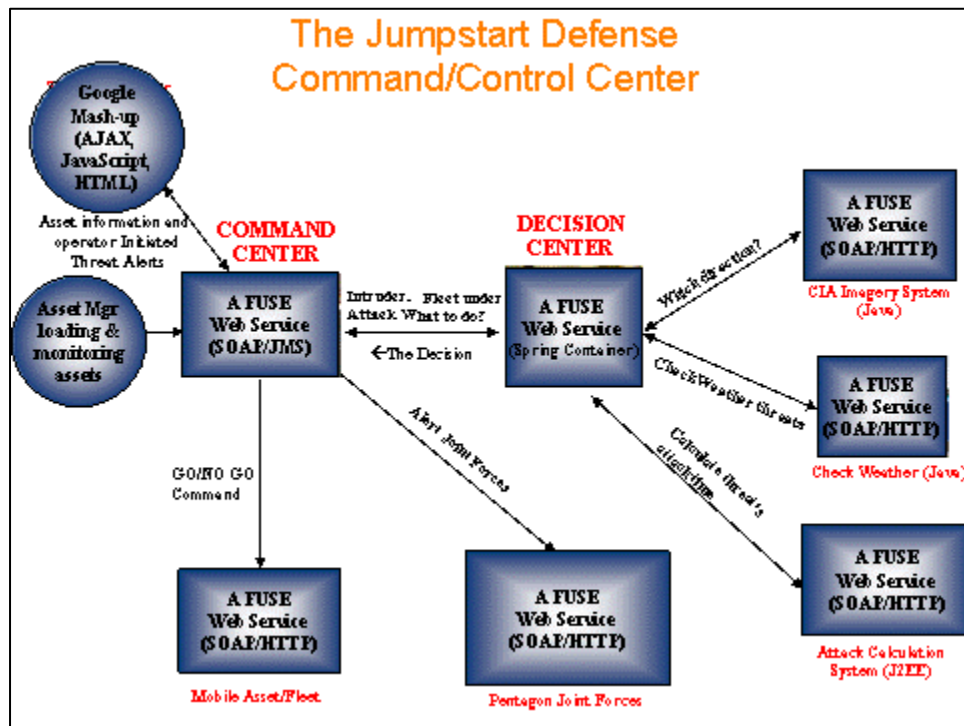
The FUSE Services Framework is a pluggable service framework based on the Apache CXF project, and it provides the easiest environment available for Java developers to create Web services. The FUSE Services Framework is the only services framework that fully implements the JAX-WS 2.0 specification, radically simplifying the process of exposing existing Java code as a Web service or writing new Web services. The FUSE Services Framework enables developers build high-performance Web services that are scalable, secure, and robust in minutes, freeing them to concentrate on the core aspects of their applications.

The FUSE Services Framework makes it easy to create new and existing services across heterogeneous environments by providing design-time tools and runtime infrastructure that are technology-neutral. The pluggable and extensible architecture allows the FUSE Services Framework's small footprint to work in a variety of container servers, with a variety of languages, and with several messaging systems.



## FUSE Mediation Router

The FUSE Mediation Router is a powerful solution for message routing based on the Camel project at the Apache Software Foundation. The FUSE Mediation Router is a powerful tool for routing and process mediation that combines the ease of basic POJO development with the clarity of the standard Enterprise Integration Patterns. The FUSE Mediation Router gives developers a tool for process design that's quite simple to use, but very powerful. It's designed to be used with message brokers like the FUSE Message Broker, within the FUSE ESB or with the FUSE Services Framework or as a stand-alone framework using one of its many transports.



# Installing the Jumpstart Appliance

The Jumpstart appliance has an install mechanism that copies the files to your system. The following summarizes the prerequisites and install process:

## **Prerequisites**

- Windows XP or Windows Vista
- 3 Gig of available disk space
- Since this is a preconfigured VMware application to minimize system installation and configuration, you will need to download and install the VMWare Player (or VMWare workstation). It is available for free at: <http://www.vmware.com/download/player>
  - The following instructions are based on VMWare Player 1.0.3 for Windows.
  - Later versions of VMWare Player e.g. 2.0 work as well.
- Internet access is required in order to access Google Maps.
- Familiarity with Java programming and JAX-WS.

## **Starting with a DVD**

Run the Install utility (autorun.inf) or execute the file .\Disk1\InstData\NoVM\install.exe. This requires a EULA acceptance then copies the files from the DVD to your hard drive.

## **Downloading the image from the web**

If you are not starting with a DVD, the image can be downloaded from the web at [www.iona.com/jumpstart](http://www.iona.com/jumpstart) . Contact your local IONA sales rep for more information.

Once you have the compressed image file named **FUSEJumpstart2.0.7z**, you need to use the 7-zip utility to unzip. You can get the compression software at: <http://www.7-zip.org/>.

Extract this file to a location on your disk.



# Running the Jumpstart Appliance

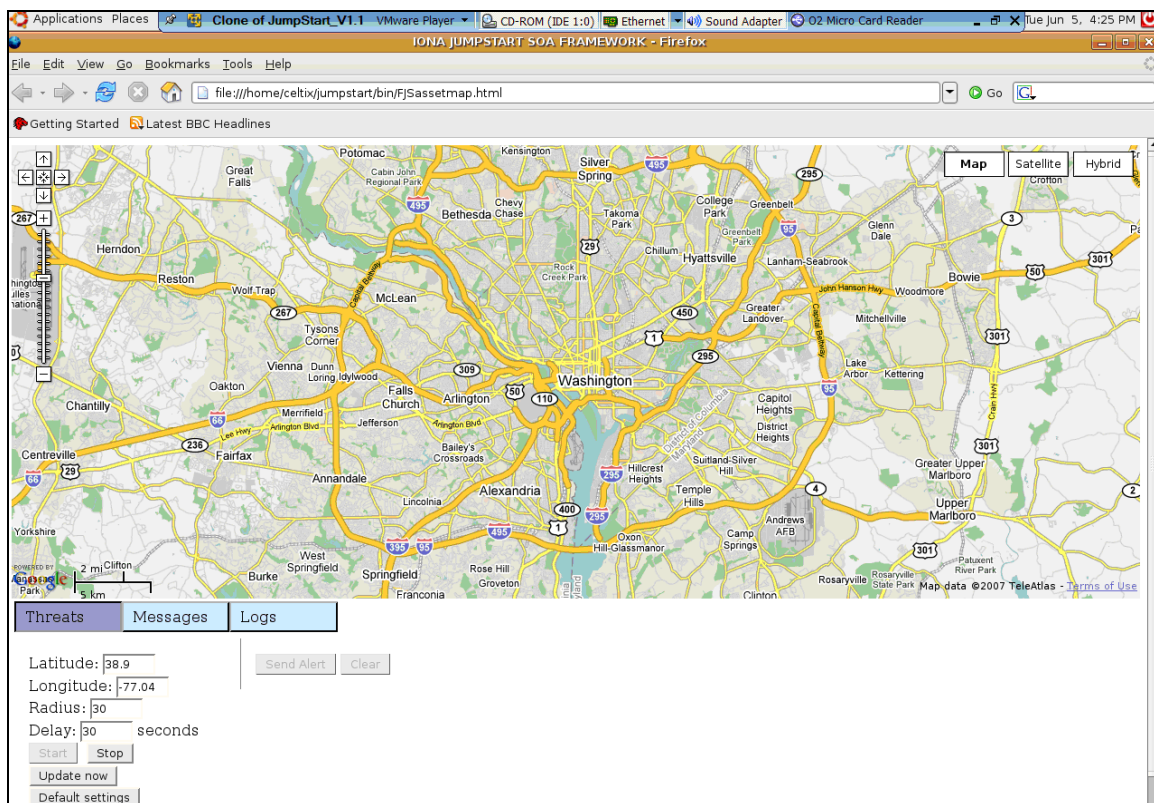
1. Start the VMWare Player and select the VMWare virtual disk file named **FuseJumpstart2\_0.vmx** from the location on the file system that the image was copied. You might receive a message stating “The Location of the virtual machine’s configuration file has changed”. If so, select “Create” to create a new identifier, then “OK”.

The system will take about 30 seconds to initialize as it loads the Ubuntu linux operating system.

- You will automatically be logged into the *iona* user account.
- The jumpstart\_user\_guide.pdf (this document) will appear.

2. Start the Firefox browser by selecting **Applications/Internet/Firefox Web Browser** or **selecting the Globe in the menu bar**.

- The browser is configured to use the following URL as the default page:  
<file:///home/iona/jumpstart/bin/FJSassetmap.html>. It will take about 5 seconds to initialize





3. Using the applications menu, start **three** terminal windows via **Applications/Accessories/Terminal**. The environment will be set automatically in each window via a sourcing of the `.bashrc` script. The `.bashrc` script sources the following scripts:
  - The `setenv.sh` script under the home directory.
  - The core FUSE environment script under (`$HOME/fuse-services-framework-2.0/bin/fuse_env.sh`) to set the base FUSE environment.
  - The core jumpstart FUSE environment script (`$HOME/jumpstart/bin/fjs_fuse_env.sh`) to set the base FUSE application environment for the jumpstart framework application.
  - The core jumpstart application environment script (`$HOME/jumpstart/bin/fjs_env.sh`) to set the jumpstart application environment.
  - Automatically puts the user into the `$HOME/jumpstart/bin` directory
  - The framework includes `.bat` files for the windows platform as well as the Ubuntu Linux platform.
  - These files can be ignored while on the Linux platform.
4. In one window, run the start services script (`./start_service.sh`). This starts the SOA services running on FUSE.
  - You should see Spring Container Ready starting and eventually reporting that it is ready. Log messages will flow to this window while the services are running when assets are reporting locations or the browser GUI is running.
5. In another window, run launch the assets script (`./launch_assets.sh 20 20`). This will put four pre-configured assets on the map.
  - You can add more by uncommenting other assets in the asset launch file.
6. At this point the SOA application is up and running. The asset driver window will report sending asset movement updates, while the spring container window will report receiving asset movement updates from the asset drivers. The asset drivers run in the background as processes while their log output goes to this window.
  - You can change the speed/update rate of the assets when launching them, by changing the input parameters. The first argument is speed, the second is update rate. For example, "`./launch_assets.sh 20 20`" identifies that the assets will move at 20 MPH and will announce or update their position every 20 seconds.
  - You may minimize all three windows at this point.
7. You should see four icons on the map. You may need to allow for the asset processes to complete their startup before they appear.
  - One platoon signifying the "Good guys"
  - Two tanks signifying field assets
  - One "Rocket" signifying the "Bad guys"
  - You may need to zoom out initially to see all the icons on the map until they move closer to the Good guys
    - The zoom control is on the upper left hand corner of the map



Latitude:

Longitude:

Radius:

Delay:  seconds

The table is locked for editing. To resume updating, click on "Send Alert" or "Clear"

Attacked	Threat	ID	Type	Lat	Lng
<input type="checkbox"/>	<input type="checkbox"/>	Badguy	Rocket	38.9721	-77.0374
<input type="checkbox"/>	<input checked="" type="checkbox"/>	East	Tank	38.8974	-76.5152
<input type="checkbox"/>	<input type="checkbox"/>	West	Tank	38.8974	-77.5596
<input type="checkbox"/>	<input type="checkbox"/>	Goodguy	Platoon	38.9042	-77.0374

8. Observe the icons moving on the screen. They will move to converge on the Good guys at a rate of 20 MPH, with an update rate of 20 seconds.

9. Acting as the operator, you will see the three buttons, **Threats**, **Messages**, and **Logs**

- The **Threats** button will display the threats panel below the map
  - The threat panel consists of settings on the left and an asset table
  - The **Latitude/Longitude** reflect the center of the map
  - The **Radius** reflects the area from the center that is monitored for inclusion in the asset table
  - The **Delay** controls the frequency in which asset information is gathered/updated by the GUI
  - The **Start/Stop** buttons control the GUI updates
  - The **Update** now button forces an immediate GUI update
  - Descriptive help screens will appear on the left as required.
- The **Messages** button will display the messages panel below the map
  - This will show "Urgent" messages when a Bad Guy is within the radius that is monitored.
  - These messages can be cleared using the Clear button



- The **Logs** button will display the messages panel below the map
    - This will show SOA (SOAP/JAX-WS) message traffic between the GUI and the various SOA services
    - These messages can be cleared using the Clear button
10. Acting as the operator, click the Threats button to bring up the Threats panel.
- Select the Badguy from the table, and select one of the other assets to create a "threat alert" that will be sent to the Decision Center from this Command Center
  
  - Click the **"Send Alert"** button on the bottom of the page. This will send the alert message with the logistics information about the pair you selected. You should see a dialog popup box. Click OK to close the dialog box.
    - The resulting displayed decision coming from the decision center will toggle between "Engage" and "Leave area" based on the contrived logic of the Decision Center SOA service
  - You should see the messages in the spring container windows showing message traffic as well as the logs panel.

**To close the demo down:**

- Open the third terminal window
- Stop the assets by using the stop\_assets script (./stop\_assets.sh)
  - This will kill the assets driver background processes
- Stop the SOA services by using the stop\_service script (./stop\_service.sh)
  - This will stop the FUSE container services and their SOA implementation instances
  - You can ignore the messages about log4j appenders



## Extending the Jumpstart Framework Use Case

The use case is designed to serve as a framework and sandbox to learn about building a SOA. The existing services can be modified, reused, or new services can be added, or services can be deleted. The `/home/iona/jumpstart` directory contains all the source files, e.g. you can edit (using VI) `CommandCenter.wsdl` located in the `src` directory.

The `build.sh` and `clean.sh` convenience files can be used to invoke the build system, which is based on `Ant`. The Java 5.0 JDK has also been installed. When adding a service, users should be familiar with XML, XML Schema, WSDL, and general principles of JAX-WS (or POJO annotations). To modify a service, only Java programming knowledge is required.











From this point on you can:

1. View the WSDL and become familiar in it's content
2. Modify an existing service
3. Create a new service
4. Utilize and become familiar with the FUSE tools
5. Change the service or the binding component of existing services
6. Progress to downloading FUSE onto your computer and perform services development

VI and command line tools are provided; however, you can load your favorite IDE into the Jumpstart appliance, e.g. if you are familiar with Eclipse, you can download it from <http://www.eclipse.org/downloads>.

## FUSE Services Framework 2.0 Documentation

The following table contains links to the online document for the FUSE Services Framework. It is also available at: <http://open.iona.com/documentation/>.

Book	HTML
Release Notes	
Writing <b>WSDL</b> Contracts	
Developing Applications Using <b>JAX-WS</b>	
Developing Distributed Applications with Dynamic Languages	
Using the <b>FUSE</b> Services Framework <b>JMS</b> Transport	
Developing RESTful Services	
Using the <b>SOAP</b> Binding	
Using the <b>XML</b> Binding	
Using the <b>HTTP</b> Transport	
<b>FUSE</b> Services Framework Command Reference	



## FUSE Services Framework Command Reference

From [http://open.iona.com/docs/command\\_ref/index.html](http://open.iona.com/docs/command_ref/index.html)

### I. Code Generation Commands

- [wsdl2java](#) — generates JAX-WS compliant Java code from a WSDL document

### II. WSDL Generation Commands

- [java2wsdl](#) — generates a WSDL document from a JAX-WS compliant Java class
- [xsd2wsdl](#) — generates a WSDL document containing the types defined in an XML Schema document.
- [wsdl2soap](#) — generates a WSDL document containing a valid SOAP/HTTP endpoint definition based on a portType element.
- [wsdl2xml](#) — generates a WSDL document containing an XML binding based on a portType element.
- [wsdl2service](#) — generates a WSDL document containing a valid endpoint definition from a binding element.

### III. XML Validation Commands

- [wsdlvalidator](#) — validates a WSDL document

## Jumpstart Source Directory Structure

In the IONA/Jumpstart/src/com/iona/federal directory are the following folders:

- **Asset** – helper code for assets to register their information
- **Attack Calculator** – implementation code for attack calculator service
- **CIAMagerySystem** – implementation code for mock imagery agency service
- **CommandCenter** – implementation code for maintaining asset information and integration with decision center
- **CommandCenterTypes** – complex datatype helper code
- **DecisionCenter** - – implementation code for orchestration of series of events which result in a decision
- **Mobilefleet** – implementation code for communicating decision to the mobile fleet service
- **Pentagon** - implementation code for communicating decision to the pentagon service
- **Util** – helper code for parsing arguments
- **Weather** – implementation code for mock weather agency service



## Create a HelloWorld Example

Reference <http://open.ionapro.com/docs/jaxws/index.html> for an example of creating a HelloWorld WSDL contract using FUSE. Additional examples using JAX-WS are also provided.

## Adding other FUSE Components and Open Source Products

- Visit <http://open.ionapro.com> to install other FUSE components, including the FUSE Message Broker, FUSE ESB, and the FUSE Mediation Router.
- For BPEL Orchestration, there are several open source alternatives including, [Intalio](#)

## For More Information and Support

- More information on IONA success in Government, <http://www.ionapro.com/solutions/government>
- For support on this Jumpstart, please contact:
  - Robert Kilker, [robert.kilker@ionapro.com](mailto:robert.kilker@ionapro.com)
  - Michelle Davis [michelle.davis@ionapro.com](mailto:michelle.davis@ionapro.com)

## Removing the Jumpstart Appliance

As the SOA Jumpstart appliance is non-intrusive, just delete the directory containing the files installed.



## About IONA

IONA's commitment to open source software is part of its 15-year heritage of solving the most complex integration problems by applying open, standards-based solutions. An industry leader in integration and SOA, IONA has served federal, state, and local government agencies around the world with its standards-based products in support of highly secure, mission-critical applications and systems. IONA has the proven expertise to design a highly flexible, distributed SOA infrastructure for the Global 2000 using standardized components. IONA customers built the first generation of SOA with its standards-based CORBA technology in the 1990s. IONA has extended that heritage with Artix, its advanced SOA infrastructure suite, and its leadership and involvement in a broad array of open source SOA projects

IONA understands that open source is more than a new model for software distribution – it is a new model for software development that can bring hundreds or thousands of developers together to pursue a common product vision. Open source is also a software development methodology that brings IT architects and product developers closer, in an open forum. IONA employs a number of leaders and participants in the open source projects that are most critical to its open source offering, and it engages with the users of its open source technologies in its community at <http://open.iona.com>.



Making Software Work Together™