



The Lightweight and High Performance ESB, Free and Open Source!

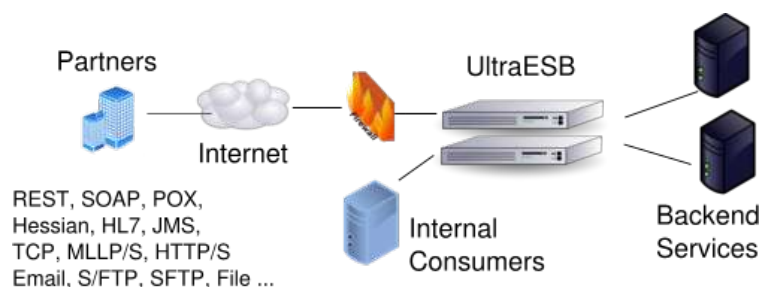
Quick Introduction

Integrate different systems both within an organization and beyond

The UltraESB is a Free and Open Source Enterprise Service Bus [ESB] that facilitates the integration of different systems; both within an organization and beyond. Integration is facilitated via "messages" which may be HTTP/S messages (such as SOAP, REST, JSON, XML, Hessian, AS2, HTML, Binary, Text etc) or messages over many other transports such as JMS, Email, TCP, MLLP/S etc, or Files FTP/S, SFTP etc. Messages may carry different types of payloads such as SOAP, XML, Text, CSV, EDI, HL7, JSON, Maps etc., and the UltraESB can accept messages over one transport in one format, and forward it to another system over another transport and another format.

Messages passing through the UltraESB can be “mediated” via fragments of Java code or JSR 223 Scripting languages such as Ruby, Groovy, Javascript and many many others as listed at <https://scripting.dev.java.net/>

Thus the “mediation” logic could be easily written from within a Java IDE the user is familiar with, and test, execute and debug the configuration all from within the IDE and utilizing the IDE features such as step through debugging.



Acting as a gateway between your organization and your partners

By deploying the UltraESB between your partners, clients and users, you can use it to validate messages received, verify security, perform decryption, filter requests for injection attacks etc, and then route it to one or more of your internal services - possibly over different transports. The UltraESB can alternatively transform the message payload using XSLT, XQuery, Milyn, Java, Groovy or any other type of transformation - or use your favorite external library or custom code you've developed to perform a transformation. It can also perform load balancing and fail-over between multiple backend systems, using round robin, weighted, random and other types of algorithms. For example, you could receive SOAP messages over HTTPS, and direct them to a "Text"ual JMS service backend; or poll an FTP location for a file pattern, process the records when a file appears, and post them to your HL7 backend systems etc.

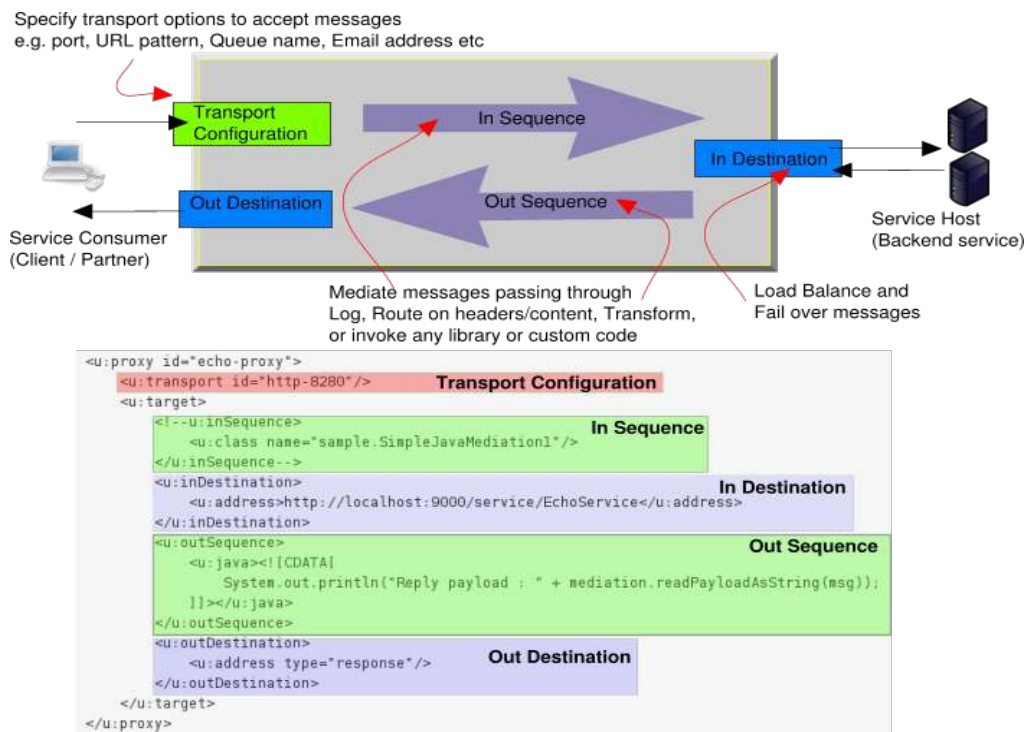
By routing all of your external traffic through the UltraESB, you could apply security consistently (e.g. WS-Security) and efficiently - without handling it in your internal code (e.g. PHP etc), mask out sensitive information such as credit card numbers consistently, use multiple identity certificates to communicate with different business partners, transform message formats or switch transports etc.

Quick Facts

- The UltraESB is an extremely light weight (~ 35MB) Java application and can be installed over any Java 1.6 supported OS/HW or VM/Cloud configuration
- The UltraESB is configured with a single XML file - or multiple files if desired, for a large configuration
- The configuration is a standard Spring configuration, and thus offers native Spring support during "mediation" of messages
- The "mediation" logic maybe written as a Java code fragment (No compiling, Jar/bundling or deploying - just write and run) or any JSR-223 scripting language such as Groovy, Ruby, Javascript etc
- Supports JTA transactions - including suspend/resume, message split/aggregation
- The UltraESB can be deployed as a Cluster, and uses Apache ZooKeeper, and is able to share and cache information across cluster nodes using ehCache
- One or more instances can be easily managed and monitored via the Web based Administration – UConsole, or any JMX based monitoring system such as Zabbix.

Proxy Services

Proxy services are the basic units of deployment over the UltraESB. A proxy service maybe exposed over one or more transports. The most basic and typical proxy service will specify information about the transport/s over which the service would listen to messages, and then specify an optional "In Sequence" that will specify "what to do" with messages received. A Sequence maybe specified as a Java class, code fragment, JSR-223 script, Spring bean etc. and will expose the message as a special variable "msg" along with another variable "mediation" that will expose a variety of mediation utilities. After an "In Sequence" a message may be forwarded to an "In Destination" if specified; if the "In Sequence" did not explicitly route the message to another location. If the forwarded endpoint issues a response message, it will be handed over to the "Out Sequence", and then the "Out Destination". The "Out Destination" typically is a response destination for HTTP/S messages, and writes the response back to the client connection.

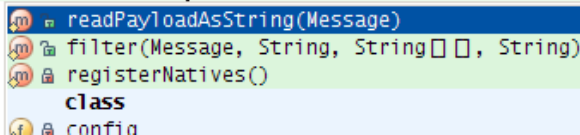


The above diagram depicts a graphical representation of a proxy service, and its definition in the UltraESB configuration file. In the above example the "In Sequence" is specified as a [temporarily commented out] Java Class and the "Out Sequence" as a Java source fragment within the configuration. These sequences are optional, and only required if you want to "mediate" the message between the client and the destination you are forwarding to. A Java fragment sequence is compiled into the memory and executed as byte code, and the user does not need to compile, bundle or deploy mediation code for these sequences.

Sequences also support JSR-223 language scripts - such as Groovy, Ruby, Javascript. In these cases as well, the script is compiled into byte code with the JDK 6 support for JSR-223 and executed optimally. In this example the "In Destination" forwards the request message to the "Echo Service" - which simply echoes back the message, and the "Out Destination" directs the response to the client. The outSequence above uses a simple System.out.println() call to print the payload to the console - however in real life you can invoke any of the bundled mediation utilities for logging, transformation, content based routing etc, or invoke any Java code or Spring bean - making the [possibilities](#) limitless. The Mediation API's and the other available configuration options are documented at <http://api.adroitlogic.org>

The Proxy service is defined within an extended Spring syntax, and can be easily edited intelligently with IDEs such as IntelliJ IDEA, Eclipse and NetBeans. Sequences such as the "In Sequence" above, written as a Java Class allows line-by-line step through debugging with the IDE (learn more about [IDE integration](#) and checkout the Screen-casts)

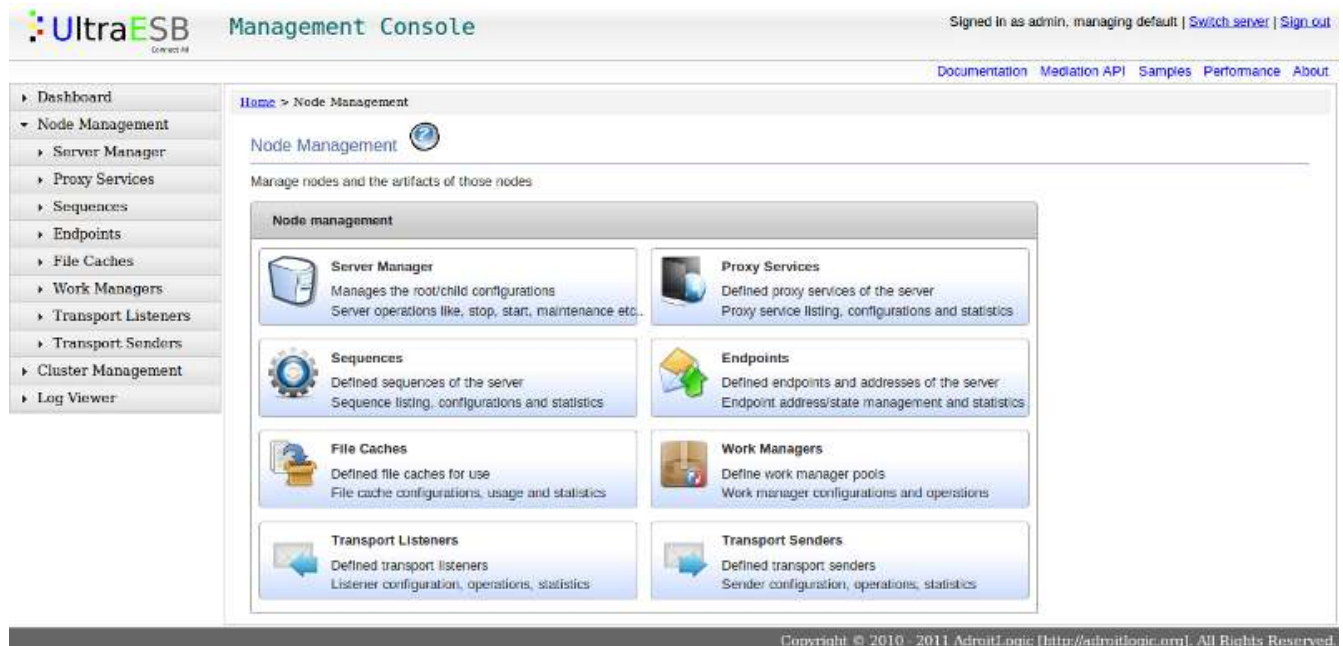
```
<u:outSequence>
  <u:java><![CDATA[
    System.out.println("Reply payload : " + Mediation.readPayloadAsString(msg));
  ]]></u:java>
</u:outSequence>
<u:outDestination>
  <u:address type="response"/>
</u:outDestination>
```



A proxy service may specify load balancing and fail-over options for messages routed to backend services; and begin, suspend, resume, commit, and rollback JTA transactions, perform error handling and recovery, split and aggregate messages etc among other things. Please see samples, and documentation for more details.

Starting the Web based Administration Console – Uconsole

After starting the UltraESB using the bin/ultraesb.sh (or .bat), start the UConsole using the bin/uconsole.sh (or .bat). The UConsole binds only over the local network interface by default, unless the interface is updated on its uconsole/conf/jetty.xml. Point your browser at <https://localhost:8043/uconsole> to manage any UltraESB instance. The UConsole detects locally executing UltraESB nodes by default and lists them. To connect to any other instance, specify the JMX URL, and the credentials.



Links to documentation

Screen-casts

Check out the [AdroitLogic channel on YouTube](#) for videos on how to configure your IDE, or start using the UltraESB. This is a must see for any new user, and provides a wealth of information in just a few minutes.

The User Guides

The User Guides are for beginners to get familiar with using the UltraESB. It introduces the user with a simple example, and helps users configure their favorite IDE to edit the configuration - with intelligent schema aware and language aware IDE support. Of course, its perfectly possible to use the Notepad, VI editor or any other text editor as well, but an IDE will help validate the configuration and pop-up possible options being aware of the context.

- [UltraESB Users Guide - Installing and Starting the UltraESB](#)
- [UltraESB Users Guide - The HelloESB Sample](#)
- [UltraESB Users Guide - Configuring and Using the IDE](#)
- [Getting started with the AdroitLogic ToolBox for the UltraESB](#)

Reference Guides

The reference guides describes details about transport, mediation and endpoint configuration and will be useful when you want to implement your own scenario if it differs from the samples you select to begin with. However, the samples can show you working configurations and using the tools to play with them allows one to quickly learn the basics as well as the details.

- [Basics of a Proxy Service](#)
- [Getting started with Endpoints or Destinations](#)
- [Getting started with Sequences and Mediation](#)
- [Reference Guide - Mediation and Configuration](#)

API Documentation for Mediation

This is the definitive guide on the methods exposed to manipulate a message, read or write its headers, payload, or properties etc, transform, write, log, report etc. These methods maybe invoked from the "Sequences" - that maybe written as Java code fragments, Java classes, Spring beans or any JSR-223 Scripting languages such as Groovy, Ruby Javascript etc. Note that the "Sequences" are automatically compiled into byte code irrespective of the source language, and thus for example even a Javascript fragment can start, commit or rollback JTA transactions etc as desired!

The API documentation also includes the configurable options for transports, WS-Security and AS2 handling.

The Javadoc API's can be found from <http://api.adroitlogic.org>

Links to Samples

In practice many users find that the samples are close to their existing requirements, and thus serves as an easy way to get started. For qualified customers we provide evaluation and POC support free of charge - and will implement a sample illustrative scenario you require at no cost to you.

Always check the site for more samples, and refer to the samples/conf directory for annotated sample configurations, though associated documentation may not yet be available for these samples.

Support

Use the [Community Site](#), and the [issue tracking system](#) to raise questions, suggest features and contribute back improvements. To share any confidential information about a problem that you cannot post publicly, email us at help@adroitlogic.com

To checkout the complete source code, use the Mercurial command:

```
hg clone https://bitbucket.org/adroitlogic/ultraesb
```

AdroitLogic offers free support for qualified customers to help get started, or implement a proof-of-concept.

Production support, Training, Consultancy, Development Support and Custom development at very reasonable rates are provided by AdroitLogic. Please inquire from info@adroitlogic.com or refer to the website <http://adroitlogic.org> for more details.