

CLEO - Client Emulation & Operations Program

Requirements Document

Table of Contents

1. [Introduction](#)
2. [Assumptions](#)
3. [General Requirements](#)
4. [Program Control and Configuration](#)
5. [Publishing Information Objects](#)
6. [Subscribing to Information Objects](#)
7. [Creation of Information Objects](#)
8. [Metadata Search \(Query\)](#)
9. [Display Metadata \(Data Returned\)](#)
10. [User Interface](#)
11. [Local Data Server Requirements](#)
12. [Definitions/Glossary](#)
13. [Standards](#)

1. Introduction

This document is intended to provide the requirements specifications for the software system known as CLEO. This software system is intended to provide a testing capability for the Joint BattleSpace InfoSphere (JBI). This project is sponsored and will be developed at the Air Force Research Laboratory.

The CLEO development effort is intended to provide the user with capabilities to publish, subscribe, and query capabilities to support the testing and evaluation JBI prototypes as well as operational systems when they become available. This goal addresses the need articulated by the technical director of the JBI Team to build a system that can be universally used to validate both functional and performance capabilities of candidate JBI platforms to support the evaluation of technologies in building this next generation system. More information is available in section 1.1 The CLEO Project, below.

1.1 The CLEO Project

- [Project Objective](#)
- [Guiding Principles](#)
- [Project History](#)

1.2 Intended Users of CLEO

The users of CLEO Version 1 consist of a limited spectrum of JBI data and information users. These users include, but are not limited to, commanders, warfighters, intelligence analysts, and especially test personnel at all levels, and research scientists.

It should be recognized that the limited base of users described above implies a limited range of connection protocols will be used in accessing the system. This implication infers that the system requirements needed to support the users are provided from the base of World Wide Web (WWW or W3C) and TCP/IP technologies. Throughout this document, various levels of WWW/W3C or TCP/IP technology to be supported will be identified by reference to the various versions of the appropriate Standard.

1.3 An Overview of the CLEO System

- [Design Elements](#)
- [Conceptual Design Diagram](#)
- [UML Use Case Diagram](#)
- [UML Class Diagram \(not finalized\)](#)

1.4 CLEO System Functions (known)

Following is the list of basic functions the CLEO system must provide.

- Cross Platform capable
- Fully Compliant with the JBI Common API (<http://www.infospherics.org>)
- Command Line invocable with parameters for automated execution from scripts
- Will use an XML Configuration file for control (Cmd-Line specified when necessary)
- Actions of modules will be controlled from the XML Configuration file using a sequence notation for timing/command identification control which will be used in the log file for denoting source of actions recorded
- Time stamp and log all activity (CSV format) including errors and exceptions
- Able to simultaneously exercise multiple JBI Platforms.
- Able to provide status of operations through an control panel interface (GUI) or logs
- Simulate 200 clients publishing, subscribing, and querying in user-defined patterns
- Provide a mechanism by which the testing can be paused either arbitrarily or conditionally
- Able to synchronize events based on time, number of operations performed or conditionally within the scripting language
- Able to initiate external programs from within the scripting language
- Able to delay operations arbitrarily or based upon a randomly seeded number or synchronizing event
- Create and submit XML-based schema for JBI Information Objects
- Create payloads and metadata to be used in creating JBI Information Objects (on the fly at times)
- Ability to exercise the JBI Platform with invalid metadata, payloads, and other invalid operational behavior to provide simplistic functional testing capabilities.
- Ability to dynamically alter metadata of JBI Information Objects that are being published to support the robust testing of the brokering functionality (distributing subscriptions) This is known as metadata iteration.
- Ability to publish JBI Information Objects from previously created 'sets' of objects available for this purpose.

- Publishing JBI Information Objects using user-defined parameters.
- Subscribing and Receiving JBI Information Objects using user-defined parameters.
- Querying and Receiving JBI Information Objects using user-defined parameters.
- Control the number of Publishers, Subscribers, and Queryiers
- Control the size of metadata and payload of the JBI Information Objects discretely.
- Control the rate of Publishers, Subscribers, and Queryiers independently or using a user-defined model.
- Publishing, Subscribing and Querying JBI Information Objects simultaneously in order test performance/scalability.
- Able to specify matching success rate for subscriptions and queries (a priori knowledge of published/ing data)
- Able to specify predicate complexity for Subscriptions and Queries
- Graphics User Interface must be CUA (Common User Access) compliant, allowing either keyboard or mouse operationality

2. Assumptions

The following assumptions were made in the preparation of this document:

2.1 User Capabilities

2.1.1 User Defined

"User" is defined to be any of the intended users of the system, as described in Section 1.2.

2.1.2 Assumptions Regarding Capabilities

- The intended users of the system have access to a workstation able to run the CLEO and access the JBI Platform.
- No other software is required to execute the CLEO except possibly a run-time (or a virtual machine).
- The CLEO system will not preclude the use of other programs or applications via multitasking (may have deliterious effects)

2.2 Local JBI Platform Server Capabilities

2.2.1 Local JBI Platform Server Defined

"Local JBI PlatformServer" is any existing JBI platform server that has a functioning JBI Common API interface capability (Minimum version level 1) as well providing the necessary management commands to allow (via properly configured security credentials) access using Common API functions. Providing local management command control to accomodate administrative control over the JBI Platform Server to establish and reset baseline conditions and data access control.

2.2.2 Assumptions Regarding Capabilities

- Each Local JBI Platform Server involved in the testing has an fully implemented JBI Common API capability.

- Each Local JBI PlatformServer has an adequate computer hardware and software environment to support both local system control and the testing load anticipated

3. General Requirements

3.1 Standards Compliance

The system must comply with all standards mandated by the Federal government for on-line transfer of metadata and data, to the best ability of available technologies. Industry and/or de facto standards will be utilized in cases where there are no applicable Federal standards, or no available implementations of Federal standards.

3.2 Metadata Content and Format Standards

The metadata content and format standards adopted by this project are those set forth by the W3C for XML and XML Schema respectively, as implemented and extended by the AFRL Joint Battlespace Infosphere (JBI) Project.

3.3 Metadata Keyword Standard

The appropriate DOD or Federal Government standard will be used as a standard keyword list when developing metadata. Other specific keyword lists may be used if the keyword(s) are inadequate for the particular type of data being described.

The DOD or Federal Government standard is subject to change.

3.4 Mime Data Standard

The W3C Mime standard will be used as a standard file format when developing payload(s). Other specific payload formats may be used if the Mime Standard is inadequate for the particular type of data or is inadequate for maintaining minimum performance requirements.

The W3C Mime Data Standard is subject to change.

3.5 Use of Technologies

The system will utilize appropriate Internet technologies in order to provide the best product for the testers. These technologies include, but are not limited to:

- W3C (WWW) and Internet related technologies, including XML, XML Schema, MIME, SSL, HTTP, HTML, and CGI.
- Languages facilitating rapid development and cross-platform capability, such as JAVA, PERL, Python, TCL/TK, etc.,
- Object-oriented design and development
- Apache, Tomcat, J2EE, JSP, JDBC, JMS, JBOSS, etc.,

There is no specific requirement that mandates that the program/system be either monolithic (having one executable), or that it even be written in a single computer language. The over-riding principles in the development of this program should be performance always exceeding the system under test, flexibility, maintainability, and cross platform capability in that order. For more information see [Guiding Principles](#). Information about the various standards and programs can be found in the Standards Section 13.

3.6 Consistency of User Interface and log files

The User Interface for the system must provide a consistent look and feel to the user. To ensure this consistency, the following requirements are made:

- All user interface elements to be presented in any format must comply with the CUA (Common User Access) standard. (One of the requirements identified by the CUA is the ability to operate any program using either keyboard or mouse independently or together.)
- All command line invoked elements must use the 'getopt' compatible switch specification. (Unix-like command-line switches)
- All log files must be compliant with CSV (Comma Separated format) or well formed XML.

3.7 Evolutionary Implementation

All major CLEO Version 1 functions (Cfg Manager, IOCreator, Publisher, Subscriber, Queryier, Control Panel, Data Retriever, Common API Interface etc..) must be modular and relatively independent, with well-defined interfaces, so that the functions can be implemented and executed, as much as is feasible, independently or simultaneously in order to provide flexibility for testing purposes as well as evolving the CLEO system for the future. In the case of object-oriented development, the modular and independent requirement for each of the main functions can be implemented with abstract classes supporting extension and polymorphism so that the classes can be extended beyond their original incarnations both now and in the future relatively easily.

3.8 Non-Functional Requirements

3.8.1 Configuration Management Plan

This should be a project brief that includes programming language(s) and version(s) used, programming environment(s), system requirement(s), runtime requirements (libraries - versions like Jini, JMS, .NET, etc.), application server(s) used and version(s), API(s) and version(s), database(s) type and technology, source code CVS location and version, technologies used (most important is combinations of section 3.5), and the version of this document used for functional requirements.

3.8.2 User and Installation Manuals

Two manuals should be provided to prospective users as part of the software distribution, a user manual and an installation manual. The user manual should provide directions for the user(s) to support all functionality that the system provides. The installation manual (or document), should contain all of the information necessary to perform an installation given the software distribution and the installation manual which should be provided with the software distribution. These manuals (or documents) can be provided electronically using the Adobe 'PDF' electronic document standard.

3.9 Backwards Compatibility

The CLEO program should be backwards compatible to any previously created version for supported of the XML configuration file and control parameters as well as the support libraries and APIs where possible. (It expected that some features of specific APIs or libraries will be deprecated by their authors and this eventuality is not within the control of the progenators.)

4. Program Control & Configuration

4.1 Test Controls

4.1.1 Control Parameters.

The following list of control parameters represents the major parameters which have been identified as principle requirements for testing purposes. All control parameters should be settable from the primary configuration file as well as from command line interface. The control parameters should be able to be saved into 'configuration sets' (files) that can be invoked using the command line or retrieved and saved from the control panel. The Configuration Manager will provide the user with an interactive configuration control tool.

4.1.2 Specific Control Parameters

Here are the primary control parameters expected to be included in the CLEO control panel:

- JBI Platform(s) to test
- Configuration Set Filename
- Number of Clients (0-200)
- Number of Connections (0-N)
- Number of Publishers, Subscribers, and Queriers (0-N, As a percentage of total amount cumulatively)
- Rates of Publishing, Subscribing, Queries (boundary conditions, constant, varying, randomized, mathematical distribution (Gaussian, ordered, etc..))
- Number and Type of JBI Information Objects Published, Subscribed to, Queried for, Archived, and Deleted.
- Size and Complexity of JBI Information Object Metadata being Published, Subscribed, or Queried.
- Size of Payload of JBI Information Object being Published, Subscribed, or Queried.
- Specify delays between successive operations and connection establishment.
- Specify JBI Information Object 'sets' and their types to be used for publishing, as well as metadata 'sets' for subscribing, and querying.
- Specify an element in the XML metadata to iterate the data contents of, and specific type of iteration to be implemented when publishing. (An example of iteration would be for an integer data type would be incrementing, decrementing, or randomly changing its value)

This list is not exhaustive, it is expected that new control parameters will be identified as test plans evolve.

4.2 XML Configuration File

4.2.1 XML Configuration File Structure

The configuration file will follow a modularized organization which correlates with the software modules so that the configuration file can be readily understood by humans and easily maintained in relation to the module that a particular section is associated with.

There will be a one to one correlation between the element names of the control parameters, the CLEO control panel names and their command line option switch names, in order to minimize the learning curve for utilization and confusion.

4.2.2 Control of Actions using a Sequence of Command with ID's

The configuration file will allow the tester(s) to create sequences of actions which will be identified by command ID's with this identification being sent to the log files with timestamps to denote source of actions. These sequences will be 'encoded' in the XML file using a tag element name equivalent to what is being identified as a 'Sequence' of actions to be performed at a specific point in time which can be specified by the tester(s).

Example might be:

```
<SEQUENCE ID="Tango">  
  
<TIME>T+15:02:30.5</TIME>  
  
<CMD ID="Cmd-043">Publish 'Brisbane_Set' </CMD>  
  
</SEQUENCE>
```

It is to be stressed that this is only an example, there is NO particular vocabulary currently required for the XML Configuration file, just a requirement for systematic and hierarchical organization and consistency.

4.2.3 Command Line Invocation

The Control parameters should be settable from the command line interface using the Unix-style (c language 'getopt') switch setting method. Additionally, a desirable capability would be the ability to specify a configuration set and modify parameters from that set as part of the command line invocation.

4.3 CLEO Configuration Manager

4.3.1 Configuration Manager Interface Module Defined

The Configuration Manager module of the CLEO program will exist as a stand-alone application which creates, modifies, deletes, and catalogs configuration 'sets' which the CLEO testing program uses to execute the tests. In this way, the testing program is not impacted by the additional overhead of having unused code in memory when executing. Additionally, this provides for ease in testing of the CLEO functions themselves where unit tests represent the major functions to be implemented.

4.3.2 Creating and Retrieving JBI Information Object Sets

A major function of the Configuration Manager will be the ability to create JBI Information Objects to be used in testing. This function will take inputs in regards to schema creation (type), metadata content, payload content, numbers and complexity of shemas, as well as JBI Information Objects derived from these. This function will also support the retrieval of JBI Information Objects from a JBI Platform and the ability to create 'sets' from these retrieved Objects.

5. Publishing JBI Information Objects

5.1 Sources of Data

5.1.1 Creating JBI Information Objects for publishing

JBI Information Objects that are published by the CLEO program will be either retrieved from already created sets which are stored that are available to the CLEO program or they will be created 'on the fly' by the CLEO program given explicit characteristics defined in advance in the XML configuration file. See section 4.3.2 above or section 7 below for more information.

5.2 Publishing Characteristics

5.2.1 Publishing Rate

JBI Information Objects will be published using the following mechanisms for the rates.

- Constant Rate
- Simple Rate of Change
- Variable Rate of Change based on various mathematical distribution models (gaussion, logorythmic etc...)

5.2.2 JBI Information Object Size

- The size of JBI Information Objects to be published will be controllable using the following factors.
- Schema and by inference metadata complexity (both in terms of number of elements and content)
- Size (in bytes) of metadata and payload content (including specifiable content)

5.3 Logging Results

5.3.1 Timestamp format

All activity which is logged will have a timestamp of the following format at the beginning of every line in the log file representing any event being logged. It will be a timestamp which includes both date and time of the format: Year (in four digits), followed by slash, followed by the Month (in two digits), followed by a slash, followed by Day (in two digits), followed by a single space, followed by the time in this format: Hours (in two digits using military format), followed by a colon (:), followed by the Minutes (in two digits), followed by a colon (:), followed by the Seconds (in two digits), and

optionally; followed by a period (.) followed by the milliseconds.

Some would represent it this way: "YYYY/MM/DD HH:MM:SS.MILLI"

Here is an example: "2003/05/02 13:37:24.003"

5.3.2 Events to be logged

The system will provide event logging for the following actions:

- Schema addition
- Sequence Creation and Destruction
- Published Object (and Elapsed time from submission to receipt of acknowledgement if explicit)

6. Subscribing to Information Objects

6.1 Source of Subscriptions

Subscriptions can be drawn from the source data sets of JBI Information Objects used for publishing. By using sets of already created JBI Information Objects, it will be possible to know in advance the success ratio of a subscription(s) for a given data set. It is when the published JBI Information Objects are created 'on the fly' by the CLEO program that this becomes slightly more problematic. This is one of the reasons that the CLEO program/system needs to operate at a speed exceeding that of the unit under test by significant margin, so that it can not only create and publish information objects fast enough to keep 'ahead' of the platform but that subscriptions can also be created *in advance* of the publication of the object needed to fulfill the subscription.

However, by specifying the constraining characteristics of published objects that are being created 'on the fly', subscription success can be dictated in advance without needing to pass the metadata of newly created objects to the subscription module 'just in time'.

Subscriptions will be created from:

- Existing (JBI Information Object) data sets using matching based on metadata content.
- Existing (JBI Information Object) data sets using Type, Element, or parameter of the metadata schema itself.
- 'On the fly' using matching based on metadata content.
- 'On the fly' using Type, Element, or parameter of the metadata schema itself.

6.2 Subscription Characteristics

Subscription performance testing presents a challenge about what is being measured, delivery performance or platform efficiency in searching/matching for subscriptions requiring delivery from publishing objects. If it is the latter, then the number and complexity of current active subscriptions in relation to publishing objects potentially matching becomes the pivotal feature we are interested in controlling with the CLEO program. If however we are interested in subscription delivery performance, then the metric we are interested in is time to deliver subscription results given

matches with publications.

6.3 General Subscriptions Requirements

6.3.1 Predicate Complexity

Hypothetically, the complexity and number of outstanding subscriptions have a direct bearing on the brokering function. Initially, the CLEO program is only required to provide simple predicate matching capability. It will use the XPath query format for its subscriptions. (See <http://www.w3.org/TR/xpath> for more information). One of the tests of the JBI Platform will entail requesting a an increasing number of subscriptions until the JBI Platform's performance degrades and identifying the resource which has been affected the most.

6.3.2 Content Subscription - Extent of predicate specification

TBD

6.3.3 Metadata Iteration

Dynamic metadata element iteration to support subscription performance tests where publishing data success rate potentially impacts on brokering performance.

6.4 Logging Results

6.4.1 Timestamp format

All activity which is logged will have a timestamp of the following format at the beginning of every line in the log file representing any event being logged. It will be a timestamp which includes both date and time of the format: Year (in four digits), followed by slash, followed by the Month (in two digits), followed by a slash, followed by Day (in two digits), followed by a single space, followed by the time in this format: Hours (in two digits using military format), followed by a colon (:), followed by the Minutes (in two digits), followed by a colon (:), followed by the Seconds (in two digits), and optionally; followed by a period (.) followed by the milliseconds.

Some would represent it this way: "YYYY/MM/DD HH:MM:SS.MILLI"

Here is an example:"2003/05/02 13:37:24.003"

6.4.2 Events to be logged

The system will provide event logging for the following actions:

- Sequence Creation and Destruction
- Submission of a subscription request
- Receipt of JBI Information Objects which were subscribed to

7.0 Creation of Information Objects

A JBI Information Object is made up of two constituent parts; the payload or actual data, and the metadata which describes that data. The metadata which describes the data being transported utilizes a data description language known as the Extended Markup Language (XML). (See the W3C XML information at <http://www.w3c.org/XML>.) In order to provide some order for the metadata to facilitate the process of locating and disseminating this data, a third construct defining the structure and content of these metadata files in a standard for data types is utilized that is called XML Schema. (See the W3C XML Schema information at <http://www.w3c.org/XML/schema>) The schema file then, is an XML file describing the metadata XML file which describes the data. In order to facilitate the creation, publishing, and dissemination (through subscription and query) of JBI Information Objects, there is a metadata repository which holds all of the metadata schema definition files for which JBI Information Objects can be created. New schema types can be added at any time given that the Information staff approval exists for that type.

The process therefore to create JBI Information Objects entails either finding a schema or creating a schema if one does not exist for the data 'type' which is desired to publish and then using this schema to build the metadata file describing the data and then publishing the resultant JBI Information Object.

This section delineates the expected capabilities of the CLEO program in creating, manipulating and saving JBI Information Objects, it does not specify what methodology or language needs to be utilized in order to implement this capability.

7.1 Creating and manipulating metadata and payload for the CLEO to utilize

The CLEO program will support the creation of both metadata and payloads to utilize in exercising the JBI Platform's functionality and performance. Ideally, the program would allow the user to visually create metadata and payload together in an "Information Object Builder" dialog box where the XML tags from an existing XML Schema can be used like a 'model' in order to facilitate proper structure. The 'XML Spy' program provides a good working model of a visual style which would be valuable in providing ease of use coupled with intuitive operational capability. The payload creation can be implemented by 'loading' the payload content from an external file or it can be created internally as XML or as binary data 'on-the-fly'. Subsequent versions of the CLEO may provide the capability to create, edit, and delete the XML Schema's for the metadata repository of the JBI Platform. At present, it is only required only to load and save metadata schemas and validate (or purposely invalidate) JBI Information metadata and payloads.

7.2 The creation of invalid Metadata and Payloads for the CLEO to utilize

A special but important capability is the ability to submit invalid JBI Information Objects to the JBI Platform in order to test its' handling of these scenarios. A partial list follows of the invalid attributes that should be supported;

- Duplicate start tag(s) in the metadata document.
- Duplicate end tag(s) in the metadata document.
- Use of an invalid tag(s) (not specified in the Schema for the metadata type.)
- Use of an empty element where data is specified in the Schema for the metadata type.
- Use of invalid data for an element as specified in the Schema. (This should include a number of possibilities, including attempted buffer-overflow type conditions) A systematic methodology for applying invalid data should be followed to attempt all feasible invalid data

types against the specified data type for an element field.

- Use of an empty attribute where data is specified in the Schema for the metadata type.
- Use an unquoted attribute value.
- Use an attribute where one is not defined by the Schema for the metadata type.
- Use of invalid data for an attribute as specified in the Schema (This should include a number of possibilities, including attempted buffer-overflow type conditions) A systematic methodology for applying invalid data should be followed to attempt all feasible invalid data types against the specified data type for an attribute.
- Use of a metadata document with no data at all.
- Use of a metadata document which contains elements and attributes, that are all invalid for both required and optional metadata elements and attributes.

7.3 Creating, saving, loading and scripting the use of JBI Information Object sets.

The CLEO needs to be able to repeat tests using the same data in order to create and compare a baseline against subsequent implementations for verifying and qualifying different implementations of the JBI Platform. In order to accomplish this, sets of JBI Information Objects need to be created, stored and retrieved for publication, subscriptions, and querying operations. These 'sets' of JBI Information Objects need to be representative of all of the different kinds of objects that might be transported by a JBI Platform and they also need to contain the invalid JBI Information Objects potentially as well. The following capabilities represent the functionality required to implement these features;

- Ability to store JBI Information Objects that are received either by subscription or query.
- Storing JBI Information Object 'sets' using XML with XSLT for reconstitution and compressing/decompressing 'on-the-fly'.
- No software limit on the number or size of JBI Information Objects to be stored and retrieved.
- The ability to publish JBI Information Objects from the stored 'sets'.
- The ability to create and store JBI Information Objects in a set from the CLEO's own creation methods (incl. invalid objects).

(Anticipating evolving capabilities with this functionality)

8.0 Metadata Search (Query)

Initially, the CLEO program is only required to provide simple predicate matching capability. The CLEO program queries a JBI Platform in a format that is XPath compatible. (See <http://www.w3.org/TR/xpath> for more information).

8.1 Complexity of Queries

Hypothetically, the complexity and number of outstanding queries have a direct bearing on the searching function's performance and capability. The CLEO will be designed to test all of the query capabilities of the JBI Platform and measure the response time for queries and the effects of operations upon them.

9.0 Display Metadata and Payload (Data Returned for

Subscriptions and Queries)

The CLEO needs to provide some feedback to the operator that it is indeed functioning, and that it is functioning correctly. One that this can be done is to provide rudimentary display capability for incoming subscriptions and queries to be displayed. To that end, the CLEO will have the ability for the operator to select a specific subscription or query and display the returning results in a dialog box which would allow the operator a 'view' of the data being received.

10.0 User Interface

The user interface of the CLEO is made up of two modes, a command line invoked interface that uses switches of the common Unix-like switch format (More widely known as the 'getopt' style) and a Graphic User Interface (GUI) supporting the Common User Access standard (CUA) with familiar dialog boxes and pull-down menus. These interfaces are and will be evolving in order to arrive at a relatively intuitive user interface which provides the operator with an interface which supports the fundamental concepts of the CLEO's operations in a straightforward manor.

11.0 Local Data Server Requirements

The CLEO is designed to run on small systems emulating clients connecting to JBI Platforms as well as larger systems with the ability to effectively saturate JBI Platforms with massive amounts of data in order to measure and qualify scalability of various implementations. CLEO is written in Java, and thus is cross-platform capable, running on MAC's, PC's, Workstations and larger systems. To the degree that I/O capability limits throughput for any one individual system, in order to execute scalability tests, the modus operandi requires that client systems of a JBI Platform under test be able to provide data at a rate faster then the Serving JBI Platform is able to ingest. This, then dictates the server hardware requirements for a CLEO.

12.0 Definitions/Glossary

tbd

13.0 Standards

This section provides links and references to applicable standards and best practices recognized and utilized by the Air Force Reseach Laboratory (AFRL) and specifically as a necessary consitituent part of the CLEO program as well as the JBI Platform.

W3C XML information at <http://www.w3c.org/XML>

W3C XML Schema information at <http://www.w3c.org/XML/schema>

W3C XML XPath information at <http://www.w3.org/TR/xpath>

The JBI Common API Working Group at <http://www.infospherics.org>

The Internet Engineering Task Force (IETF) at <http://ietf.org> (To retrieve RFC's)

The Apache Foundation at <http://www.apache.org> (Apache and related topics)

Sun Microsystems at <http://www.sun.com/java> (Java, Jini, J2EE, and other related technologies)

The Object Management Group at <http://ww.omg.org> (UML, MDA and Corba)

Last Update: Tuesday Sept 16th, 2003