



---

# Collaborative Infrastructure Conceptual Overview

**TM09-0001**

Version 1.3 Draft

15 January 2010

---

Printed on:

2/15/10 9:17 AM

This is an uncontrolled copy when printed.

Refer to NX for the latest version.

---

**NOTICE:** Not for use or disclosure outside of NASA Ames Research Center except under written agreement.  
© 2009 NASA Ames Research Center. All Rights Reserved.

---

**Technical Memorandum  
TM09-0001**

COLLABORATIVE INFRASTRUCTURE  
CONCEPTUAL OVERVIEW

VERSION 1.3 DRAFT

**CONTACT**

Dr. William J. Clancey – Principal Investigator (650) 604-2526

Dr. Maarten Sierhuis – Project Manager (650) 604-4917

**ABSTRACT**

This document provides a conceptual overview of the Collaborative Infrastructure (CI), a software framework for hosting distributed software components called actors. The CI framework provides a collection of commonly needed services to its hosted actors, including services for directory registration and lookup, message transport, publish/subscribe data distribution, and actor management.

**DATE:** 15 January 2010

**KEYWORDS:** Collaborative Infrastructure, Application Interface, Actor, Communicative Act, Directory Service, Transport Service, Data Distribution Service, Management Service

This document has not been reviewed by the Intellectual Property Organization.

---

Printed on:

2/15/10 9:17 AM

This is an uncontrolled copy when printed.

Refer to NX for the latest version.

---

**NOTICE:** Not for use or disclosure outside of NASA Ames Research Center except under written agreement.  
© 2009 NASA Ames Research Center. All Rights Reserved.

---

## AUTHORS

Bob Nado

Ron van Hoof

William J. Clancey

Maarten Sierhuis

---

Printed on:  
2/15/10 9:17 AM

This is an uncontrolled copy when printed.  
Refer to NX for the latest version.

---

**NOTICE:** Not for use or disclosure outside of NASA Ames Research Center except under written agreement.  
© 2009 NASA Ames Research Center. All Rights Reserved.

---

**APPROVED**

---

**Maarten Sierhuis**

Date

Project Manager

---

---

Printed on:

2/15/10 9:17 AM

This is an uncontrolled copy when printed.

Refer to NX for the latest version.

---

**REVISION HISTORY**

Version	Contact	Action	
Version 1.0 Draft 12/18/2009	Bob Nado (650) 604-0413	New	First draft
Version 1.1 Draft 12/18/2009	Bill Clancey (650) 604-2526	Add	Brief History section and additional references
Version 1.2 Draft 12/21/2009	Maarten Sierhuis (650) 604-4917	Add	Added some more history from the SAVH, ESAS 6E project
Version 1.2 Draft 1/15/2010	Bill Clancey (650) 604-2526	Minor edit	Fixed figure references
<i>Actions Taken are: <b>New</b> = new document, <b>Add/Delete/Change</b> = a section or topic has been added, or deleted, or changed.</i>			

Printed on:  
 2/15/10 9:17 AM

This is an uncontrolled copy when printed.  
 Refer to NX for the latest version.

# TABLE OF CONTENTS

<b>1. INTRODUCTION .....</b>	<b>1</b>
1.1 SUMMARY .....	1
1.2 BRIEF HISTORY .....	2
1.3 INTENDED AUDIENCE .....	3
<b>2. ARCHITECTURE OVERVIEW .....</b>	<b>4</b>
2.1 PROCESS MANAGER.....	5
2.2 ACTOR HOSTING ENVIRONMENT .....	6
2.3 ACTOR DEFINITION AND CONFIGURATION .....	7
<b>3. MESSAGING .....</b>	<b>8</b>
<b>4. FRAMEWORK SERVICES.....</b>	<b>10</b>
4.1 DIRECTORY SERVICE.....	10
4.2 TRANSPORT SERVICE .....	11
4.3 TRANSLATION SERVICE .....	12
4.4 DATA DISTRIBUTION SERVICE .....	12
4.5 MANAGEMENT SERVICE .....	13
4.6 TIME SERVICE .....	14
4.7 LOGGING SERVICE .....	14
<b>5. REFERENCES .....</b>	<b>15</b>

## FIGURES

FIGURE 1. ORIGINAL COLLABORATIVE INFRASTRUCTURE CONCEPT FROM SAVH PROJECT (ESAS 6E) .....	2
FIGURE 2 OVERVIEW OF THE COLLABORATIVE INFRASTRUCTURE ARCHITECTURE .....	5
FIGURE 3 MESSAGING SEQUENCE DIAGRAM.....	9
FIGURE 4 DISTRIBUTED DIRECTORY SERVICE ARCHITECTURE .....	10
FIGURE 5 TRANSPORT SERVICE ARCHITECTURE .....	11
FIGURE 6 DATA DISTRIBUTION SERVICE ARCHITECTURE .....	13
FIGURE 7 MANAGEMENT SERVICE ARCHITECTURE .....	14

---

Printed on:

This is an uncontrolled copy when printed.

2/15/10 9:17 AM

Refer to NX for the latest version.

# 1. INTRODUCTION

## 1.1 SUMMARY

The Collaborative Infrastructure (CI) provides an environment for hosting distributed software applications and components that provides a number of services that can be used to simplify management, status monitoring, network transport, messaging, data distribution and translation.

CI is intended to simplify how applications and components can interact with one another shielding them from most of the complications that come with distributed computing involving transport services and directory services and providing a set of general services to monitor and control these applications and components, to allow them to synchronize time and to manage a unified way of logging application messages.

CI provides a set of interfaces and services to enable application developers to easily enable their application to interface with other applications. The CI supports interactions between applications in a highly distributed environment allowing applications to find and/or discover one another and managing the transport of data between applications. The CI application interface design accomplishes this by defining a set of generic interfaces to be used and/or extended by application developers to connect their applications to the collaborative infrastructure.

Agent-based architectures have been found to be effective for implementing distributed systems that include components that need to be proactive and operate with some degree of autonomy. Agent-based architectures typically enable cooperation and collaboration among agents by providing a high-level agent communication language and support for sending messages using that language between agents. The CI infrastructure is particularly well suited for hosting agent-based software as it provides for communication among agents using the FIPA standard Agent Communication Language based on Communicative Acts [1][2].

CI has the following noteworthy features:

- uses the strengths of CORBA for defining the component interface and for defining data structures transmitted in messages
- a distributed replicating directory service supporting name-based and capability-based lookups
- a transport service with pluggable transport protocols (TCP/IP, UDP/IP, SSL (FIPS 140.2 compliant), Multicasting, ...)
- a data distribution service to support publish/subscribe and UDP/IP multicasting while hiding implementation details
- generic messaging using FIPA Communicative Acts

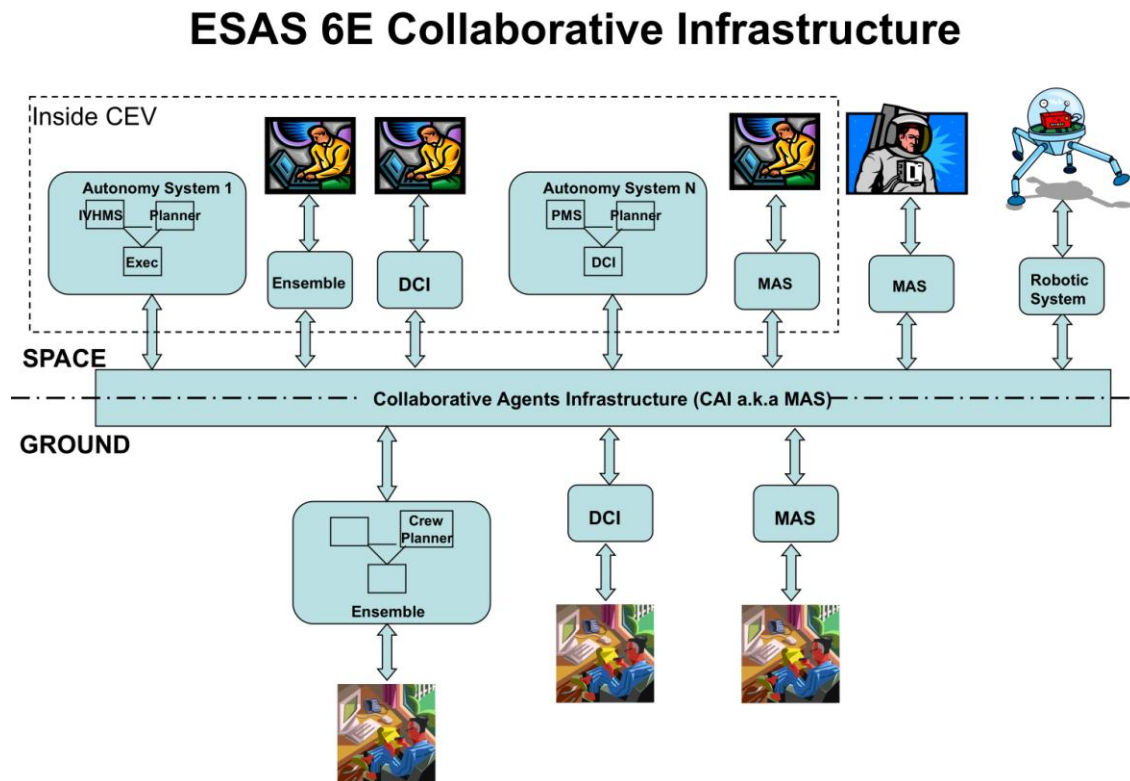
## 1.2 BRIEF HISTORY

The CI framework was formalized as part of the Exploration Technology Development Program, Task 6, “Autonomy for Operations” (A4O) FY06 and has been used by A4O for advanced development research since then.

In FY06, the project plan for the Spacecraft Autonomy for Vehicles and Habitats project, funded as part of Element 6E of the Exploration Systems Architecture Study, defined the Collaborative Infrastructure as follows:

“**Collaborative infrastructure** – handle information and request flow among automated systems, coordination software, and users.”

The original concept for the Collaborative Infrastructure (CI) is depicted in Figure 1. The idea for the CI was to develop a common middleware communications infrastructure for autonomous multi-agent systems.



**Figure 1. Original Collaborative Infrastructure concept from SAVH project (ESAS 6E)**

The CI framework was derived from the open architecture and agent-based systems integration methods developed during the Mobile Agents Project (NASA Intelligent Systems Program, 2001-2005; ETDP, Task 12, "Human-Systems Integration," 2006). ***The combination of CI with Brahms agents, referred to as the Mobile Agents Architecture (MAA), has proven to provide extreme flexibility for rapidly configuring a wide variety of distributed hardware and software systems by small development teams:***

The MAA was originally developed to support distributed interplanetary operations involving multiple robots, a vehicle like the lunar rover, voice commanding, cameras, biosensors, plans and schedules, databases, email, GPS, and other instruments and software programs, used by a crew and ground support. Mobile Agents systems were field tested at the Mars Desert Research Station (2003-2005) and Desert-RATS (2002, 2005, 2006) providing services for recording science data, navigation, scheduling, and biomedical monitoring under the rubric of "automating CapCom" for Mars surface exploration. See Clancey, et al. (2006) and [Desert-RATS 2006 video](#)).

During FY06 MAA was used within a habitat simulation to integrate an electric power system including a generator, inverter, photovoltaic panels, and batteries, enabling crew members to receive performance data, trend analyses, and alerts about the power system via a blue tooth headset (see [Power Agents video](#)).

During FY07 the MAA was used for pressurized suit partial gravity (POGO) tests (JSC Human Research Program), integrating a metabolic rate algorithm in Excel with biosensors and voice commanding). Plus it was used in FY08 in the Moon & Mars Mission Analog Missions (MMAMA) program by field geologists to collect data in simulated lunar EVAs in New Mexico and Hawaii as a single-user configuration (iMAS). iMAS was used in FY08 for coral surveys by the Smithsonian Institution using voice commanding with scuba headsets.

During FY07 – FY10, the CI has been incorporated in OCAMS, the OCA Management System for automating routine file operations between the ISS and ground support, including international partners. OCAMS, like the advanced development projects from which it is derived, automates workflow among people and a variety of distributed hardware and software systems (Clancey, et al. 2008).

## 1.3 INTENDED AUDIENCE

This document is intended for people who wish to evaluate the Collaborative Infrastructure for possible use in a system design or are interested in an overview of its benefits and services.

## 2. ARCHITECTURE OVERVIEW

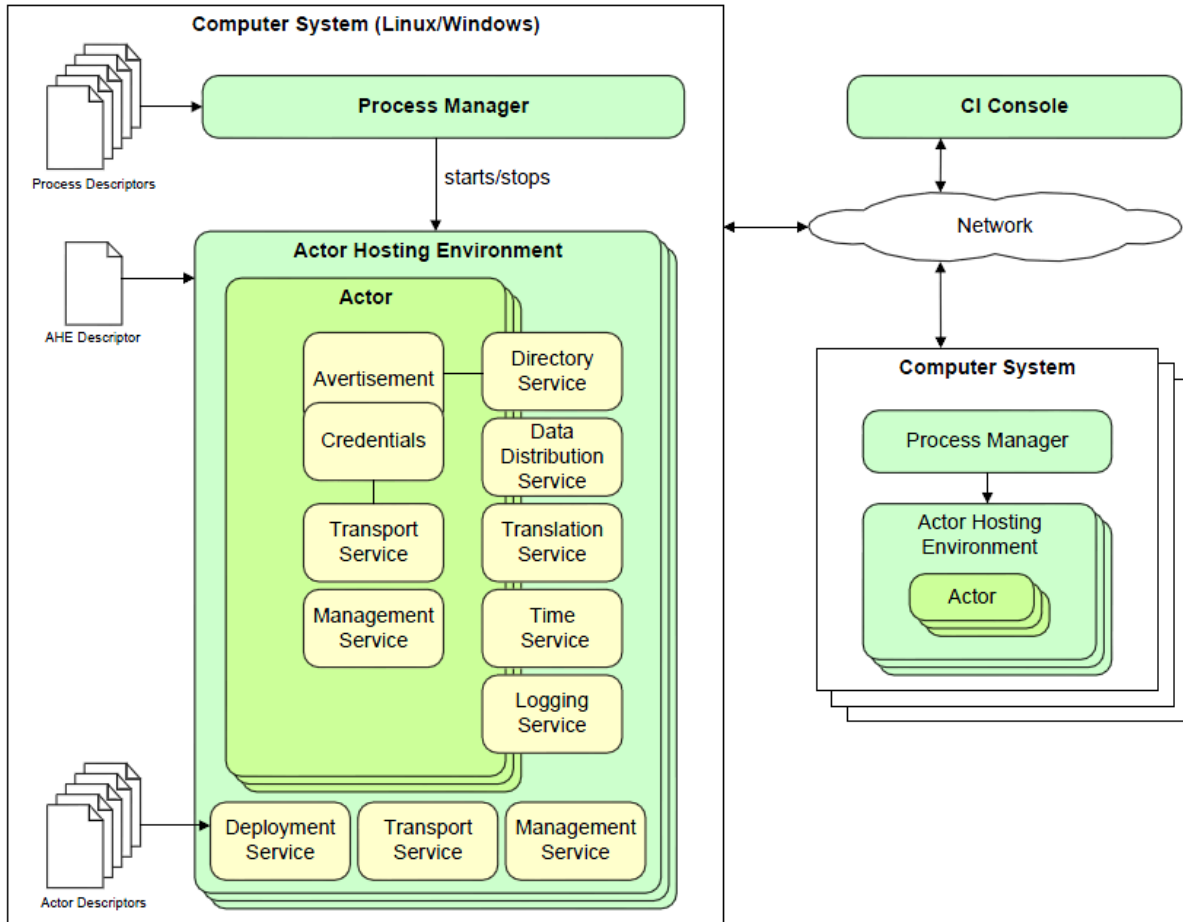
To enable software components written in multiple programming languages and distributed over multiple platforms with potentially different operating systems to collaborate, the Collaborative Infrastructure provides a higher-level framework layer over functionality provided by CORBA (Common Object Request Broker Architecture) [3] and network transport protocols.

CORBA provides a language-independent way, using its Interface Definition Language (IDL), to define the interfaces presented by each of the CI framework services. CORBA, however, has several deficiencies in the areas of directory services, transport services, and interoperability between different ORB implementations. CORBA provides only name-based directory services that suffer from single point of failure problems due to the lack of a distributed implementation with replication of directory entries. The CORBA IIOP communication protocol provides only point-to-point communication over TCP/IP with no support for the use of UDP or multicasting for data that does not need delivery guarantees.

To address these problems, the Collaborative Infrastructure includes a distributed replicating directory service supporting name-based and capability-based lookups and a transport service that abstracts away the details of different transport implementations and provides dynamic selection of the most appropriate transport protocol to use based on the message and its intended recipient.

Whereas inter-object communication in CORBA is based on the invocation of methods on objects, CI provides for actors to communicate using either a generic, agent-based messaging service or a publish/subscribe data distribution service.

Software components in CI are referred to as actors and are hosted in an Actor Hosting Environment that provides framework services and which is managed by a Process Manager (see Figure 2).



**Figure 2 Overview of the Collaborative Infrastructure Architecture**

## 2.1 PROCESS MANAGER

To simplify the management of the applications and their startup the collaborative infrastructure provides a way to easily manage and monitor CI applications.

Every computer system that is to run one or more CI applications must run one CI Process Manager. This CI Process Manager is responsible for starting and stopping CI Actor Hosting Environments. Each CI Process Manager has a configuration file in which is specified which actor hosting environments are available on that specific machine. For each actor hosting environment a configuration file is defined specifying the startup parameters for that actor hosting environment. These configuration files are published in the process manager's deployment directory where the process manager looks to find the available actor hosting environments.

The CI Process Manager publishes the list of actor hosting environments to allow for remote management applications (such as the CI Console) to select what actor hosting environments to start/stop. Generally only one actor hosting environment would be required per system, however it is possible to run multiple actor hosting environments that have different configurations to allow for parallel testing of various applications/actors.

## 2.2 ACTOR HOSTING ENVIRONMENT

Actors are hosted in an actor hosting environment. The actor hosting environment ensures that the services required by an actor are made available to it. The actor hosting environment is one system process allowing for multiple actors to run in one process allowing them to share services such as the directory service, transport service, data distribution service, time service and logging service.

An actor hosting environment is a process in which one or more actors can be loaded, managed and unloaded. The actor hosting environment provides the CI related services to the actor. Some services are shared by all hosted actors, while other services are created and provided to an actor on a per actor basis. The hosting environment determines this depending on the type of service.

Each actor hosting environment is itself a special actor and is configured using a configuration file. This configuration file specifies the deployment directory used by the actor hosting environment and specifies which actors should automatically be loaded. The deployment directory contains the configuration files of all the actors that can be loaded in the actor hosting environment. The actor hosting environment can publish a list of the available actors to allow for a remote management application to select which actors to load without having to require the actors to be specified in the hosting environment's configuration file. The actor hosting environment publishes methods to load and unload actors from the actor hosting environment.

## 2.3 ACTOR DEFINITION AND CONFIGURATION

Every actor is defined in either a shared library (C++) or a Java archive file (Java) and must implement an IActor interface that defines a number of methods required by the CI framework. The CI provides an abstract class with implementations of many of the IActor interface methods, in particular methods for accessing the various services. This abstract class may be directly extended by an actor implementation or serve as the basis for a class to which the actor implementation delegates IActor interface method calls. The actor implementation is responsible for methods that query the actor's state and for initialize, start, suspend, resume, stop, reset, and suspend methods. In addition, the concrete actor class must implement a processMessage method to handle incoming messages.

Actors are uniquely identified by their Credentials specifying their name(s) and any other identifying information to allow other actors to identify the actor and verify that the actor is indeed the actor they are expecting to interface with. The Credentials encapsulate information about how actors can interface with the actor for which the Credentials are given. This information is captured by endpoint objects. Each actor can support multiple endpoints allowing for multiple ways to communicate with the actor depending on the needs of the actor. The endpoints specify a protocol used to communicate with an actor, host and port information, as well as any additional information required by the protocol. The endpoints are used by the transport service to establish communication links between two or more actors to transmit messages/data between the actors.

An actor may be configured using a configuration file specifying the credentials, advertisement and transport related properties for the actor as well as specifying where the file in which the actor is defined can be found. The actor's configuration file must be stored in an actor hosting environment's deployment directory to allow the actor to be loaded and started. The actor hosting environment uses the configuration file to properly load and configure an actor and its services.

### 3. MESSAGING

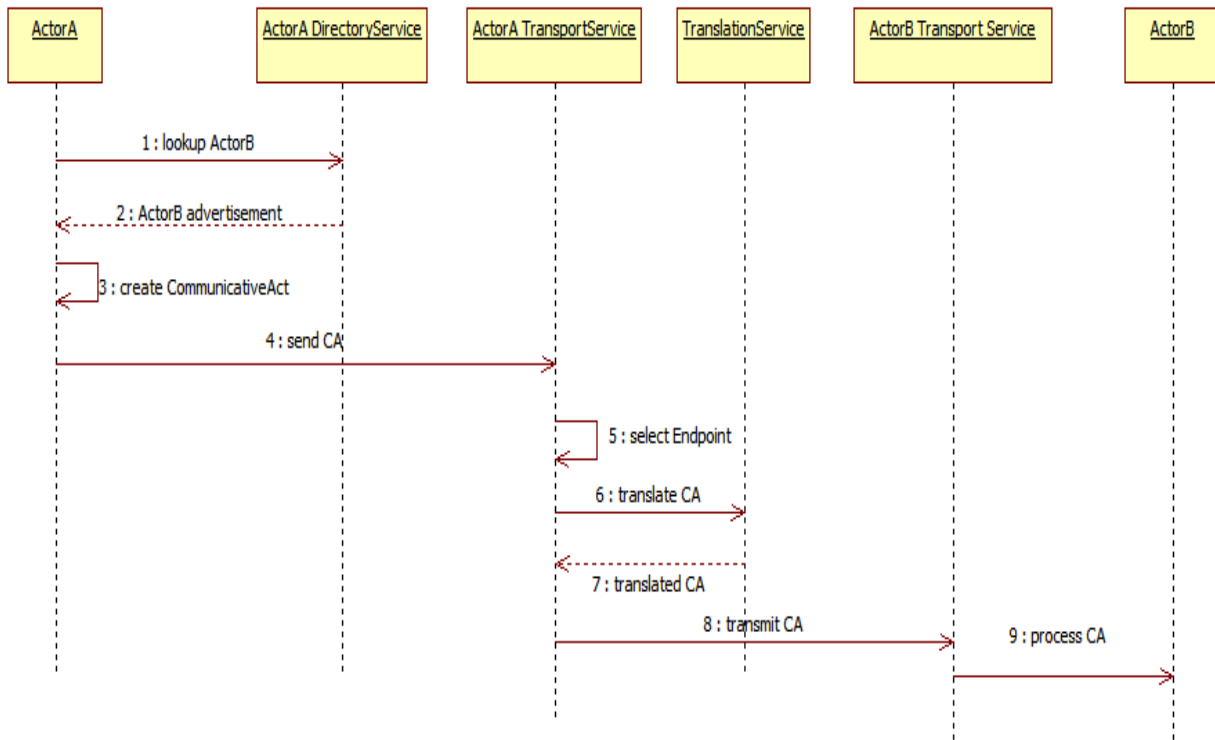
The principle mode of communication between actors connected to the collaborative infrastructure is message based (point to point). A message based communication layer was chosen to allow for a pluggable transport layer supporting multiple transports, to simplify the interactions between actors hiding the complexities of communications between actors, and to easily support quality of service parameters and security parameters without impacting the interfaces of the actors themselves.

The messages that are communicated between actors are called Communicative Acts. A communicative act defines a message that is based on the communicative act standard [1,2] defined by FIPA (Foundation of Intelligent Physical Agents). A communicative act consists of an envelope with addressing information (sender, receiver, replyTo) and transport hints and consists of a payload for the message content and content related parameters. Fundamentally, all messaging is asynchronous in CI. If an actor expects an immediate response, the transport service provides a method to synchronously send a message and then block waiting for a response.

Figure 3 displays a UML sequence diagram illustrating how ActorA sends a message to ActorB in the CI framework. ActorA must first obtain the credentials of ActorB from the directory service provided by its hosting environment. ActorA invokes a lookup method on the directory service providing a suitable name for ActorB and receiving ActorB's advertisement in return. ActorB's advertisement may be cached so that the directory service need not be accessed for every message that ActorA sends to ActorB. ActorA next creates a Communicative Act object with addressing information, the content of the message, and properties describing security and quality of service parameters. The newly created Communicative Act is then sent to ActorA's transport service. The transport service examines ActorB's credentials to find the endpoints that ActorB supports. A particular endpoint is chosen based on security and Quality of Service properties specified in the Communicative Act. The transport service also examines ActorB's credentials to determine the Communicative Act languages that it supports. If the language for the content in the payload used in AgentA's Communicative Act does not match one of the supported languages, the transport service will invoke the actor hosting environment's translation service to try to find and execute a translator between the source language and one of the recipient's languages<sup>1</sup>. Finally, ActorA's transport service will use protocol information from the selected endpoint to transmit the possibly translated Communicative Act to ActorB's transport service. When received by ActorB's transport service, the transmitted Communicative Act is reconstructed and passed to the processMessage method of ActorB. The implementation of ActorB may process the received Communicative Act in any desired manner, but the CI framework provides support for dispatching the Communicative Act to message handlers based on the performative and action of the Communicative Act.

---

<sup>1</sup> Translation can happen at either the source or at the destination of the message depending on the availability of translators on either end.



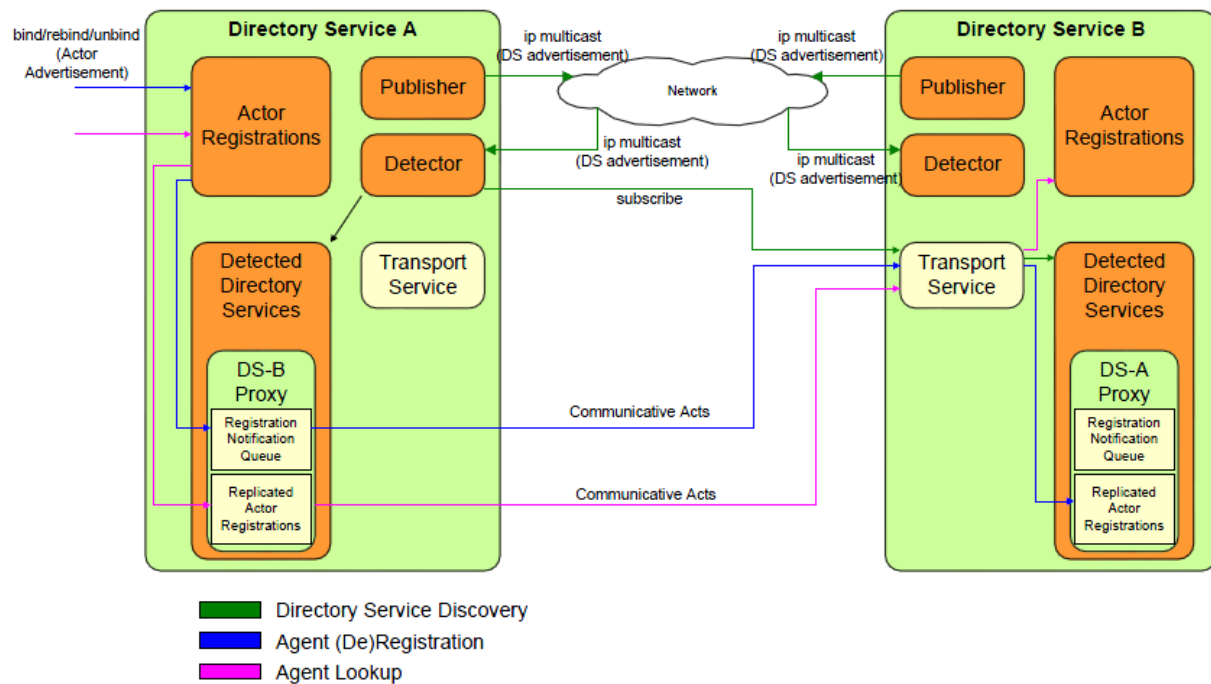
**Figure 3 Messaging Sequence Diagram**

# 4. FRAMEWORK SERVICES

## 4.1 DIRECTORY SERVICE

CI provides a distributed directory service. Each actor hosting environment has its own local directory service which discovers other directory services on the network and replicates directory entries between the discovered directory services. The directory service communication is implemented using the CI Data Distribution Service (see [Section 4.4](#)) and uses that service for discovery of other directory services.

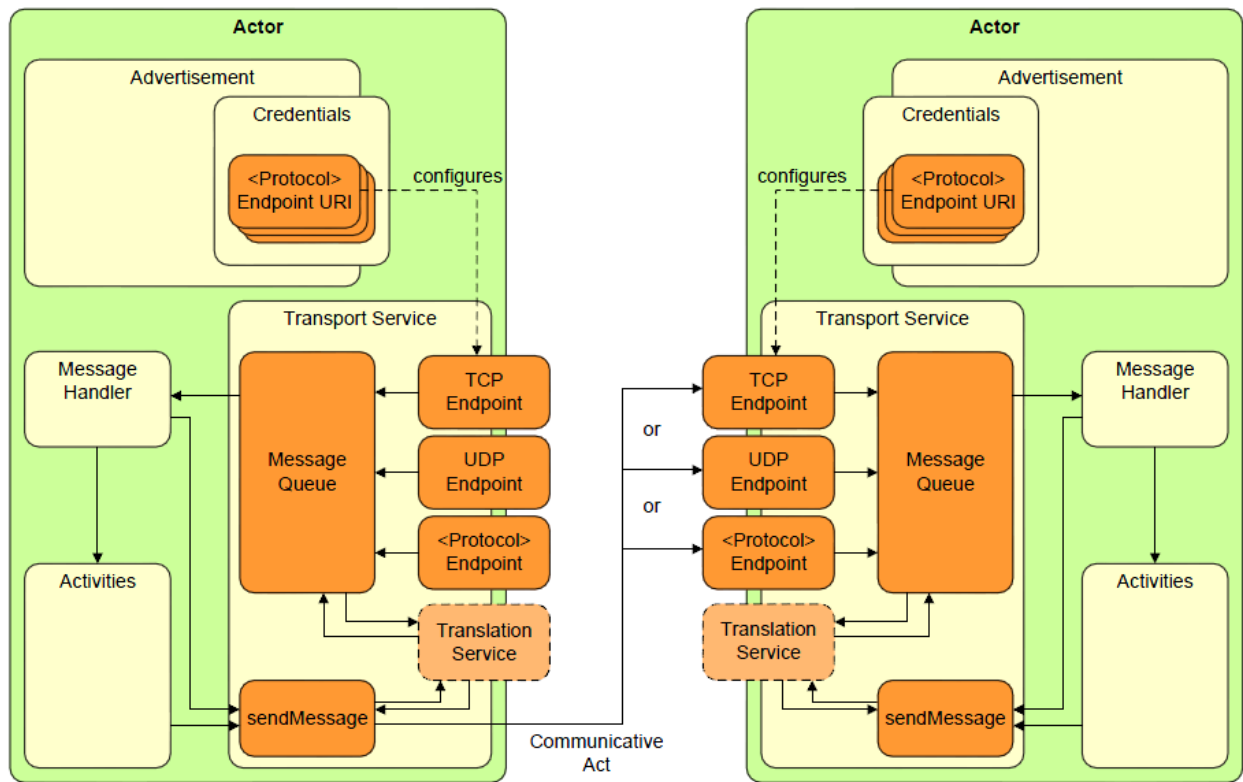
An actor is published by means of an Advertisement describing the credentials, capabilities and other properties for the actor. Advertisements are registered in the directory service. When other actors need to locate actors by name or by capability they search for those actors using the directory service. The directory service returns the advertisements for those actors matching the search parameters.



**Figure 4 Distributed Directory Service Architecture**

## 4.2 TRANSPORT SERVICE

An actor uses the transport service to send messages (communicative acts) to other actors. The sending actor is responsible for creating and populating a communicative act optionally using the factory methods provided by the transport service. The transport service selects the most appropriate transport protocol based on the end points published by the intended recipients, transport hints provided in the message and the Quality of Service (QoS) and security related parameters specified for the message. The transport service also provides services to create messages.



**Figure 5 Transport Service Architecture**

## 4.3 TRANSLATION SERVICE

As part of an actor's configuration file one or more languages can be defined to indicate in what languages the actor can process messages without requiring translation and to indicate the preferred language. A transport service (either at the sending or receiving end of the message transmission) uses this information to process a message before sending it on its way to the intended recipient actor. If the language of the message does not match one of the languages accepted by the recipient actor the transport service will ask the translation service to try to locate a translator between the source and target languages.

## 4.4 DATA DISTRIBUTION SERVICE

In addition to message-based communication between actors, CI provides a data distribution service (DDS) that takes care of managing the publication and subscription of data. An actor can register as a publisher and/or as a subscriber of data that has a particular named domain and topic.

CI hides the particular implementation used by the DDS. QoS properties associated with the data to be published allow the service to select the most appropriate implementation for the data distribution. The service can opt to use communication based on protocols such as TCP/IP, UDP/IP, SSL, or multicast. The various CI DDSs communicate with one another about the data that is being published and the data for which they have subscribers allowing the CI's data distribution services to optimize their data distribution opting not to distribute data at all if no subscribers are present.

The DDS provides three means for a local DDS to discover other DDSs on the network: multicasting, using DDS tables, and by means of a DDS coordinator. A DDS table is a listing of all other DDSs that the local DDS needs to coordinate with. A DDS coordinator distinguishes a single DDS with which all other DDSs coordinate. The DDS coordinator is more flexible in that discovery of new DDSs is possible; with DDS tables this is not possible since it is a fixed set.

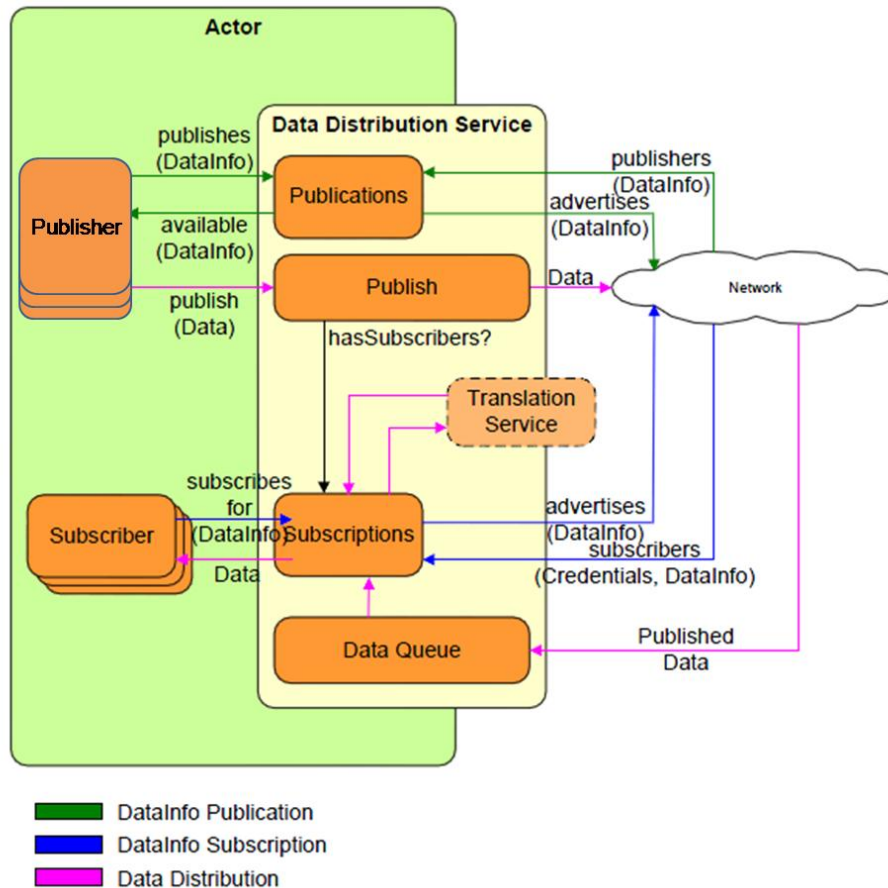


Figure 6 Data Distribution Service Architecture

## 4.5 MANAGEMENT SERVICE

The management service is used as a central point of access for monitoring and management tools to monitor the health of an actor, to monitor managed attributes, and to control actor state and configuration using managed operations and attributes. Managed operations typically include start, suspend, resume, stop, etc. Actors can be configured to have their management service periodically publish a “heart beat” with actor status information using the CI Data Distribution Service.

Every actor has its own management service that is published as part of its advertisement. Management tools such as the CI Console only interact with the management services of the actors so as not to interfere directly with the general operations of the actor.

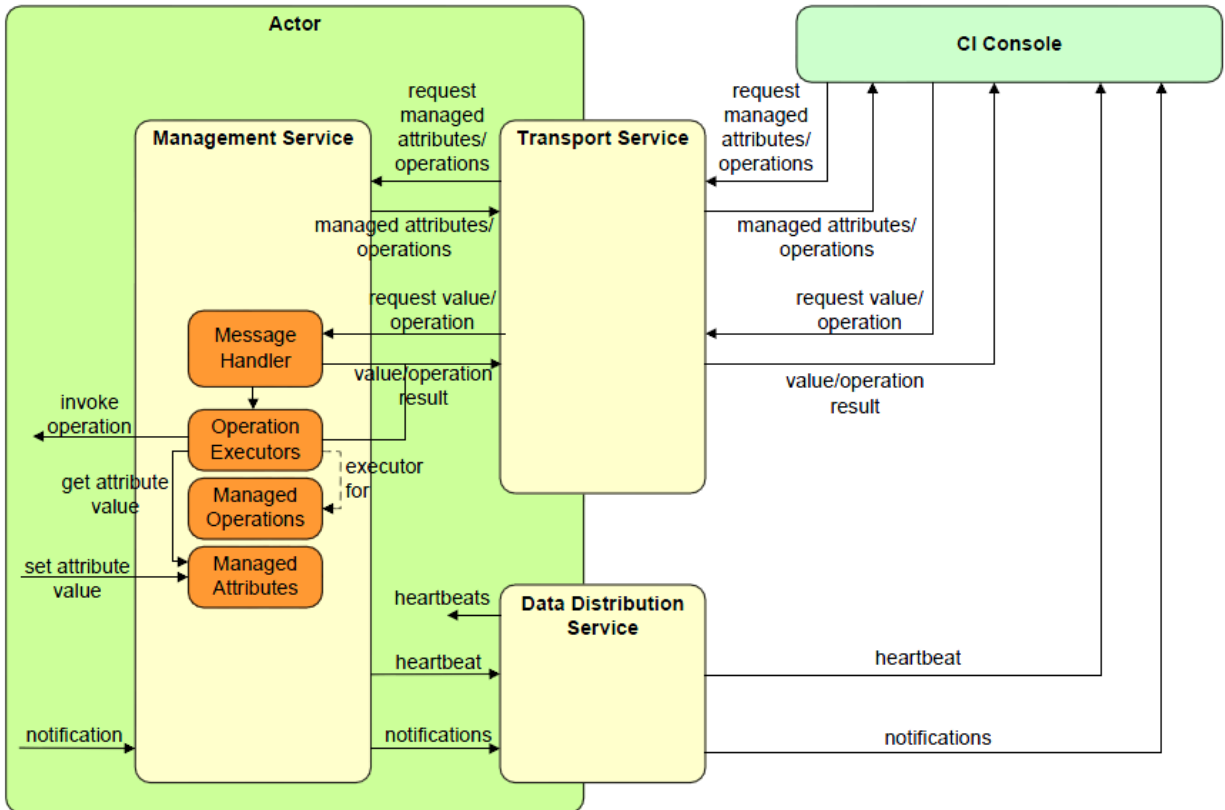


Figure 7 Management Service Architecture

## 4.6 TIME SERVICE

The time service provides actors access to date and time information from a time synchronized source ensuring that all the actors use the same date/time base. An actor can register with the time service to receive periodic notifications of the current date/time.

## 4.7 LOGGING SERVICE

The logging service is a generic interface to log various types of messages varying from debug messages to fatal messages allowing for standardized logging and storage of these messages.

## 5. REFERENCES

1. Clancey, W.J., Sierhuis, M., Alena, R., Berrios, D., Dowding, J., Graham, J.S., Tyree, K.S., Hirsh, R.L., Garry, W.B., Semple, A., Buckingham Shum, S.J., Shadbolt, N. and Rupert, S. (2005). [Automating CapCom using Mobile Agents and robotic assistants](#). *American Institute of Aeronautics and Astronautics 1st Space Exploration Conference*, 31 Jan-1 Feb, 2005, Orlando, FL.
2. Clancey, W.J., Sierhuis, M., Seah, C., Buckley, C., Reynolds, F., Hall, T., Scott, M. (2008) [Multi-agent simulation to implementation: A practical engineering methodology for designing space flight operations](#). In A. Artikis, G. O'Hare, K. Stathis, & G. Vouros (Eds.), *Engineering Societies in the Agents' World VIII*. Athens, Greece, October 2007. Lecture Notes in Computer Science Series, Volume 4870. Heidelberg Germany: Springer, pp. 108-123.
3. FIPA ACL Message Structure Specification, *Foundation for Intelligent Physical Agents*, SC00061G, 2002/12/03, <http://www.fipa.org/specs/fipa00061/SC00061G.html>.
4. FIPA Communicative Act Library Specification, *Foundation for Intelligent Physical Agents*, SC00037J, 2002/12/03, <http://www.fipa.org/specs/fipa00037/SC00037J.html>.
5. CORBA, Wikipedia, [http://en.wikipedia.org/wiki/Common\\_Object\\_Request\\_Broker\\_Architecture](http://en.wikipedia.org/wiki/Common_Object_Request_Broker_Architecture)