MASIF: The OMG Mobile Agent System Interoperability Facility

Dejan Milojicic*, Markus Breugst3, Ingo Busse3, John Campbell5†, Stefan Covaci3, Barry Friedman2‡, Kazuya Kosaka4, Danny Lange2‡‡, Kouichi Ono4, Mitsuru Oshima4, Cynthia Tham2‡†, Sankar Virdhagriswaran1, and Jim White2

1Crystaliz, Concord, MA, USA, 2General Magic, Inc., Sunnyvale, CA, USA, 3GMD Fokus, Berlin, Germany, 4IBM, Tokyo, Japan, 5Open Group, Cambridge, MA, USA

Abstract: MASIF is a standard for mobile agent systems which has been adopted as an OMG technology. It is an early attempt to standardize an area of industry that, even though popular in the recent past, still has not caught on. In its short history MASIF has raised interest in industry and academia. There are already a number of projects pursuing MASIF reference implementation. MASIF addresses the interfaces between agent systems, not between agent applications and the agent system. Even though the former seem to be more relevant for application developers, it is the latter that impact interoperability between different agent systems. This paper describes two sets of interfaces that constitute MASIF: MAFAgentSystem and MAFFinder (the acronym MAF is used for historical reasons). MASIF extensively addresses security. The paper provides a brief description of MASIF and its interfaces, data types and data structures.

Keywords: CORBA; Interoperability; Java; Mobile agents; Security; Standards

1. Introduction

Mobile agents are a relatively new technology, but there are already a number of implementations, such as Agent Tcl [1], Aglets [2], MOA [3], Grasshopper [4], and Odyssey [5]. These systems differ widely in architecture and implementation, thereby impeding interoperability, rapid proliferation of agent technology, and growth of the industry. To promote interoperability and system diversity, some aspects of mobile agent technology must be standardised. MASIF [6] is a collection of definitions and interfaces that provides an interoperable interface for mobile agent systems. It is as simple and generic as possible to allow for future advances in mobile agent systems. MASIF specifies two interfaces: MAFAgentSystem (for agent transfer and management) and MAFFinder (for naming and locating).

The original intent for MASIF was to keep it simple for the first phase and only deal with the minimal features needed for interoperability. For example, MASIF defines parameters in the agent profile to specify the requirements the agent has on the receiving agent system. This allows an agent system to support as many agent profiles as its implementation allows. Language interoperability is just one of the parameters in the agent profile. This is not a big limitation because Java is becoming the de facto standard. Therefore, interoperability in this document is not about language interoperability. MASIF is about interoperability between agent systems written in the same language expected to go through revisions. Language interoperability for active objects that carry "continuations" around is difficult, and it is not addressed by MASIF. Furthermore, MASIF does not standardise local agent operations such as agent interpretation, serialisation/deserialisation, and execution. In order to address interoperability concerns, the interfaces have been defined at the agent system rather than at the agent level. MASIF standardises:

- Agent management. One can envision a system administrator managing agent systems of different types via standard operations in a standard way: create an agent, suspend it, resume and terminate.

- Agent transfer. It is desirable that agent applications can move freely among agent systems of different types, resulting in a common infrastructure, and a larger base of available system agents can visit.

- Agent and agent system names. Standardised syntax and semantics of agent and agent system names allow agent systems and agents to identify

---

1Currently with the Sun Microsystems, East; *Currently with HP Labs; †Currently with Cisco; ‡With IBM during the initial part of MASIF; ‡‡Currently with Freegate.
each other, as well as clients to identify agents and agent systems.

- **Agent system type and location syntax.** The agent transfer cannot happen unless the agent system type can support the agent. The location syntax is standardised so that the agent systems can locate each other.

MASIF, in its current form, provides the features required for the first level of interoperability which is the transport of agent information where the information format is standardised. Once the information is transferred from one agent system to another, how the agent system deals with the parameters internally is an implementation matter and not addressed by the MASIF standard. Such information includes an agent profile which describes the language, serialisation and other requirements the agent has on the current agent system. MASIF makes it possible for an agent system to understand the requirements the agent has of its system because we believe that it is the first step in end-to-end interoperability.

Table 1 describes the types of interoperability MASIF addresses, and estimates the complexity of agent systems required to support it. **Agent management** allows agent systems to control agents of another agent system. Management is addressed by interfaces for suspending, resuming, and terminating agents. This is straightforward to implement. **Agent tracking** supports locating agents registered with MAFFinders (naming service) of different agent systems. This is also straightforward to implement. Agent communication is outside the scope of MASIF, and it is extensively addressed by CORBA [7]. Agent transport defines methods for receiving agents and fetching their classes. This requires cooperation between different agent systems and is complex to achieve.

There are other aspects that should be standardised when the industry is more mature. The security issues become complex when an agent makes a multi-hop between security domains. Most security systems today deal only with single-hop transfer. Standardising multi-hop security should be delayed until security systems can handle the problem. Today’s mobile agent systems use different languages (e.g. Tcl and Java). The effort to convert between encodings is too complex. When the code and serialisation formats are similar, it should be possible to build standard bridges between different agent system types.

### 2. Basic Concepts

An agent is a computer program that acts autonomously on behalf of a person or organisation. Most agents are programmed in an interpreted language for portability. Each agent has its own thread of execution, so tasks can be performed on its own initiative. A mobile agent is not bound to the system where it begins execution. It has ability to transport itself from one system in a network to another. **Agent state** (execution state and the attributes) and code are transported while an agent travels. An agent’s authority identifies the person or organisation for whom the agent acts. Agent names are required for identification, management operations and locating. Agents are named by their authority, identity and agent system type, whose combination has a unique value. An agent’s identity is a unique value within the scope of the authority that identifies a particular agent instance.

An agent system is a platform that can create, interpret, execute, transfer and terminate agents. Like an agent, an agent system is associated with an authority that identifies the person or organisation for whom the agent system acts. An agent system is uniquely identified by its name and address. A host can contain one or more agent systems. An agent system type describes the profile of an agent. For example, if the agent system type is Aglet, the agent system is implemented by IBM, supports Java as the Agent Language, uses Itinerary for travel, and uses Java Object Serialization. MASIF recognises agent system types which support multiple languages and serialisation methods. A client requesting an agent system function must specify the agent profile (agent system type, language and serialisation method) to uniquely identify the requested functionality.

An agent transfers itself between places. A place is a context in which an agent executes. It is associated with a location, which consists of the place name and the address of the agent system where the place resides. An agent system can contain one or more places and a place can host one or more agents. If an agent system does not support places,