Agentcities Platform Interoperability Test Suite

Version 1.1

David Bonnefoy[*], Steven Willmott[#], Nicolas Lhuillier[*], Ion Constantinescu[#], Jérôme Picault[*], Yann Camenen[*]

[*] Networking and Applications Lab
Centre de Recherche de Motorola – Paris
Espace Technologique – Commune de Saint Aubin
91193 Gif-sur-Yvette
Paris, France

[#] Laboratoire d'Intelligence Artificielle,
Departement D'Informatique,
Ecole Polytechniques Federale de Lausanne,
LIA-DI-EPFL, IN(Ecublens), EPFL, CH-1015, Lausanne
Switzerland.

Abstract

The Agentcities project is an initiative to deploy a worldwide testbed of agent platforms and services based on the FIPA Agent standard. For platforms to check their level of standards compliance it will be necessary to provide on-line, automated testing facilities to establish a particular platforms level of standard conformance and functionality. This report details an initial draft of a set of interoperability tests for implementation in the network.

1 Introduction

This document lists a number of tests which could be carried out between platforms in the Agentcities network to verify their compliance. This will be an important part of the initial set up of the network. The tests are intended to:

- Provide a series of standard interactions with verifiable outcomes between two agent platforms (i.e. tests are pair wise only – although three way tests could be added)
- Be used by contributors to the agent network when:
    - Introducing a new platform
    - Verifying that an existing platform is still effectively interacting with others after (e.g.) an upgrade, software change, hardware change.

The tests are *not symmetric*, that is the two platforms in the test play different roles (the "tester" and the "target") with different functionality tested on each platform. For a pair of platforms to be mutually interoperable according to these tests they would need to be run twice (each platform taking each role once). Nonetheless, the tests are designed so that the specific requirements for the "target" platform are minimal, allowing it to be tested without specific developments (except a "Ping" agent).

Previous work related to this document can be found in [Charlton et. al. 00] and [Camenen00].

2 Testing Approach

The following sections give additional information about the tests and their intended application.

2.1 Important Note

The tests described here are not intended to be criteria for participation in the Agentcities network. The tests simply indicate which interoperability features can be expected to work correctly between two platforms. A platform which is able to successfully complete only a subset of tests may still play an active part in the network (e.g. a platform without DF or AMS would still be able to host agents accessing/providing Agentcities services).

2.2 Refinement

This document is a first draft for discussion. Once there is some agreement it is likely that a small number of platforms will implement the tests and attempt to refine them. Afterwards these platforms can be used in the Agentcities network as references for others to test against.

Once this phase is complete the tests could be automated using (e.g.) a servlet or similar to allow new Agentcities platforms coming on line to test themselves at any time. A further step is to offer the test sequence as an agent service itself, although this can only apply to higher levels of testing since it assumes at least message exchange is working.

2.3 Considerations

Important considerations when developing this test suite include:

1. It should be comprehensive
2. It is expected to be executed periodically but not extremely frequently (e.g. *not* executed automatically every hour)
3. It should where possible undo actions it does to ensure that neither platform involved is left in an inconsistent state. This is especially important for the "tester" platform that would be permanently running when the tests are fully automated.
4. Correct/incorrect behaviour should be clearly discernable
5. Complexity of agent services required to be running on platforms being tested – objective is to have target platform as simple as possible whereas the "tester" platform may be more complex.
6. It should be possible to drive the tests from one of the platforms and preferably from one agent to simplify synchronisation and data collection.

3 Tests

The tests are divided into several areas and cover a subset of the most basic FIPA infrastructure specifications (primarily management, messaging and some language level aspects). The platforms involved are:

- Tester Platform: this platform hosts several agents[1] which carry out a series of actions
- Target Platform: this platform hosts agents and FIPA services which are the targets of the tester agent's actions.

The tester and target will normally be distinct platforms but the tests may also be useful to execute on a single platform.

3.1 FIPA specifications

The test specification does not name specific FIPA specifications at this stage but it is expected the tests below might be carried out for the following variations where they are available on both platforms:

- HTTP and IIOP Message Transport
- S-Expression (String) and XML ACL message syntax

3.2 Requirements

Apart from the FIPA Management services (ACC, MTS, AMS and DF), the tests below require the two platforms in question to have the following agents running and available:

- Tester platform:
  - One "tester" agent (named tester@tester_platform_name) which implements the functionality required to send and receive messages associated with the tests described, receive input parameters (such ass address of the target platform) and to provide output of some kind. For the federation test the tester should also implement DF functionality.
  - One "forwarder" agent (named forwarder@tester_platform_name) which forwards all messages it receives to the tester agent.
  - One "hacker" agent (named hacker@tester_platform_name) which is able to perform modification etc. request for another agents profile (which it is not authorised to do)

- Target platform:
  - One ping agent (see annex) which responds to a QUERY-REF "ping" message with the response INFORM "alive", named:
    - ping@target_platform_name
  - There must be NO AGENT named "nemo@target_platform_name" on the target platform
  - There must be no agent named "tester@tester_platform_name" registered in the target platforms DF and AMS.
  - There is no service name "df_federation_test" registered with the platform DF

When carrying out the tests there will need to be some pragmatic decisions made on (e.g.) maximum timeouts for responses and the like to facilitate automation.

3.3 Breakdown

The tests are broken down into 5 areas and documented in the sections that follow:

I. Area I: Agent Message Transport Service

[1] It is assumed in the text that tests are carried out by agents running on a platform but in fact this may be some other standalone systems – the assumption is purely to simplify the text.

II. Area II: Protocol Management/Message Exchange
III. Area III: Agent Management Service
IV. Area IV: Directory Facilitator

3.4 Area I: Message Transport Service (MTS)

1. Send/Receive a message to/from a single agent
   - Tests: target platform ACC, message receiving, delivery and sending.
   - Action (tester agent): Sends a "ping" message to the ping agent on the target platform.
   - Expected Behaviour: The tester agent receives a response from the ping agent in question (the designated receiver). As this test is for the MTS only, any response is accepted.

2. Send a message to one agent with multiple agents in reply-to (multicast-reply)
   - Tests: target platform ACC multicast, agent correct handling of reply-to
   - Action (tester): Send a message to ping on the remote platform, naming the forwarder agent and the tester agent in the "reply-to" field.
   - Expected Behaviour: Messages from the ping agent are received by tester and by tester via forwarder.

3. Send a message to a non-existing agent
   - Tests: correct behaviour of target's ACC/AMS/Platform when receiving messages for non-existent agents.
   - Action (tester): send a message to agent nemo@target_platform_name.
   - Expected Behaviour: The correct error message is received by tester from the remote platform (according to FIPA Agent Message Transport Service Specification, it should be a FAILURE with "internal-error" as predicate)

4. Sending to messages to incorrect addresses
   - Tests: target's ACC for the ability to ignore addresses which do not work and use multiple addresses
   - Action (tester): Send a message to the ping agent on target and in the sender's AID include at least two addresses, the first of which does not lead to an agent platform.
   - Expected Behaviour: tester receives a response message from ping which has the first (failed) address stripped out of the AID for the receiver.

3.5 Area II: Protocol Management/Message Exchange

Checking these basic message-handling items may forewarn of problems which might arise in later sections which make use of longer message exchanges.

1. Usage of the "conversation-id" field in ACL
   - Tests: if the ping agent at the other end (and hopefully by extension other agents) make good use of the conversation id field of an ACL message[2].
   - Action (tester): Send a message to the ping agent using a conversation-id set to an arbitrary (but legal) value
   - Expected Behaviour: the tester agent receives a response from the ping agent which includes the same value in the conversation id field as the one it originally sent.

2. Usage of "reply-with"/"in-reply-to" field in the message
   - Tests: if the ping agent at the other end (and hopefully by extension other agents) make correct use of the "reply-with"/"in-reply-to" fields of an ACL message[3].
   - Action(tester): Send a message to ping using an arbitrary (but legal) value in the "reply-with" field

[2] Note that the FIPA spec does NOT say "if you receive a message with a conversation id field set, you should respond to that message using the same conversation id". This appears to be very useful however – hence it is tested here.
[3] In this case the FIPA spec is strict.

- Expected Behaviour: The tester agent receives a response message including the same string in the in-reply-to field.

3.6 Area III: Agent Management Service (AMS)

AMS functionality is split into two sections – i) basic functionality to ensure that the AMS is actually able to perform its function and ii) functionality to do with authentication and minimal security.

3.6.1 III.i: Basic Functionality

1. Get AP-description
   - Tests: if the target platforms AMS is able to return a complete AP description
   - Action (tester): Send message to target platform AMS asking for platform description
   - Expected Behaviour (1): the tester agent receives a complete AP-description in an ACL message.
   - Expected Behaviour (2): the description matches the configuration data available for the platform (e.g. addresses given at the beginning of the test).
   - Additional: the data in the AP description may be used in the remaining tests where appropriate.

2. Effective registration
   - Note: In this test and the following ones, a Search operation is needed to ensure that the tested operation has been performed (a confirmation by the AMS is not enough). But this means that the Search function is tested at the same time. That's why the registration and search are presented here as two separated stage: in case of failure, it will be easier to find where the problem is.
   - Tests: if the target platform's AMS is able to receive and perform registrations
   - Action (1) (tester): send a registration message to the target platform AMS using the FIPA request protocol.
   - Expected Behaviour (1) (if AMS allows dynamic registration): the AMS correctly follows the request protocol, either registers the tester OR gives a reasonable error message for refusing this particular case
   - Expected Behaviour (1) (if AMS does not allow dynamic registration): the AMS correctly follows the request protocol AND gives a reasonable error message for refusing this particular case
   - Action (2) (tester): if the registration is accepted, perform an AMS search to check that agent is now registered.
   - Expected Behaviour (2): AMS gives back a message to the tester containing its description.
   - (This test may be carried out several times to check the effect of supplying/no supplying mandatory fields in the AMS Description submitted.)

3. Modify Description [if tester is registered]
   - Tests: if the AMS correctly carries out modification actions requested by an agent already registered
   - Action (tester): For each of the fields in the AMS registration (name, language, ontology, protocol, services etc.) perform the following:
     - Send a request message to modify the item
     - Perform an AMS query to check that the modification and no others have been carried out
   - Expected Behaviour: All modifications are accepted and appear in the description.

4. Effective deregistration [only if tester is registered]
   - Tests: if the AMS correctly deregisters a registered agent
   - Action (1) (tester): send a request to deregister to the AMS on the target platform
   - Expected Behaviour (1): that the REQUEST protocol was correctly adhered to and that the AMS confirms deregistration.
   - Action (2) (tester): if a reply was received (deregistration confirmed), perform an AMS search to check that the agent was really deregistered.
   - Expected Behaviour (2): Searching for the agent just deregistered returns failure.

5. Modify AMS description on a non existent agent
   - Tests: that modify actions fail correctly if an agent is not registered

- Action (tester): Send a modify request to the AMS on the tester agent’s profile (just deregistered).
- Expected Behaviour: correct adherence to the request protocol and a failure of the action (agent not registered)

3.6.2 Area III.ii: AMS Security/Authentication

(Note that these tests are for information only – it is not clear from the FIPA specs which behaviours should be enforced.)

1. Registration (false)
    - Tests: if the remote AMS allows an agent to register another agent.
    - Action (hacker): attempt to register the tester agent in the AMS by sending a register message
    - Expected Behaviour: AMS refuses the registration (NOTE: it is not clear this is correct – maybe agents should be allowed to register other agents…)
    - Cleanup: if registration was successful – deregister.

2. Registration (Correct) [Only if the AMS accepts remote registration]
    - Actions (tester): register the tester agent with the target AMS as in the section above
    - Expected Behaviour: successful registration.

3. Try to make Illegal modifications [If tester registered]
    - Tests: if the remote AMS allows other agents to modify a registration which does not belong to them
    - Action (hacker): send a modify request to the AMS to modify tester’s stored profile. Search the AMS to check that the test has been carried out (search also IF the modification was refused…)
    - Expected Behaviour (1): the modification request is refused
    - Expected Behaviour (2): the search reveals no change in the tester’s profile

4. Illegal Registrations [if tester registered]
    - Tests: if the remote AMS overwrites existing registrations when new ones are made
    - Actions (hacker): try to register tester agent again, search to confirm no changes were made
    - Expected Behaviour (1): refusal – either due to an “already registered message” or due to the fact that hacker cannot register “tester”
    - Expected Behaviour (2): no changes made in testers profile

5. Illegal Deregistration [if tester registered]
    - Tests: if the remote AMS allows an agent to deregister another agent
    - Actions (hacker): try to deregister tester agent again, search to confirm no changes were made
    - Expected Behaviour (1): refusal of request
    - Expected Behaviour (2): no changes made in testers profile

6. Deregister [if tester registered]
    - Action (tester): deregister.

3.7 Area IV: Directory Facilitator (DF)

DF functionality is split into three sections – i) basic functionality to ensure that the DF is actually able to perform its function, ii) functionality to do with authentication and minimal security and iii) DF federation.

3.7.1 Area IV.i: Basic Functionality

1. Effective registration
    - Tests: if the target platform’s DF is able to receive registrations
    - Action (tester): send a registration message to the target platform DF using the FIPA request protocol. Registration should contain 1 or more service descriptions.
    - Expected Behaviour (1): the DF correctly follows the request protocol, either registers the tester OR gives a reasonable error message for refusing this particular case

- Action (2) (tester) : if the registration is accepted perform a DF search (using agent name) to check if the agent is now registered.
- Expected Behaviour (2): DF gives back a message to the tester containing its description.
- (This test may be carried out several times to check the effect of supplying/no supplying mandatory fields in the DF Description submitted.)

2. Service Search [If tester registered]
    - Tests: if the target platform's DF can be queried for services provided by a registered agent
    - Action (tester): send a search message to the target platform DF containing the names of services previously registered
    - Expected Behaviour: the relevant service and agent descriptions are returned (AT LEAST those registered by tester)

3. Modify/Search Description [if tester is registered]
    - Tests: if the DF correctly carries out modification actions requested by an agent already registered
    - Action (tester): For each of the fields in the DF registration perform the following:
        - Send a DF modify request message to modify the item
        - Perform a DF search query to check that the modification and no others have been carried out
    - Expected Behaviour: All modifications are accepted and appear in the description.
    - (This test may be carried out several times to check the effect of supplying/no supplying mandatory fields in the DF Description submitted.)

4. Effective deregistration [only if tester is registered]
    - Tests: if the DF correctly deregisters a registered agent
    - Action (1) (tester): send a request to deregister to the DF on the target platform, await reply
    - Expected Behaviour (1) : that the REQUEST protocol was correctly adhered to and that the DF confirms deregistration.
    - Action (2) (tester): if a reply was received (deregistration confirmed), perform an DF search to check that the agent was really deregistered.
    - Expected Behaviour (2): Searching for the agent just deregistered returns failure.

5. Modify DF description on a non existent agent
    - Tests: that search/modify actions fail correctly if an agent is not registered
    - Action (tester): Send a modify request to the DF on the tester agent's profile (just deregistered).
    - Expected Behaviour: correct adherence to the request protocol and a failure of the action (agent not registered)

3.7.2 Area IV.ii: DF Security

(Note that these tests are for information only – it is not clear from the FIPA specs which behaviours should be enforced.)

1. Registration (false)
    - Tests: if the remote DF allows an agent to register another agent.
    - Action (hacker): attempt to register the tester agent in the DF by sending a register message
    - Expected Behaviour: DF refuses the registration (NOTE: it is not clear this is correct – maybe agents should be allowed to register other agents…)
    - Cleanup: if registration was successful – deregister.

2. Registration (Correct) [Only if the DF accepts remote registration]
    - Actions (tester): register the tester agent with the target DF as in the section above
    - Expected Behaviour: successful registration.

3. Try to make Illegal modifications [If tester registered]
   - Tests: if the remote DF allows other agents to make a registration of an agent profile which does not belong to them
   - Action (hacker): send a modify request to the DF to modify tester's stored profile. Search the DF to check that the test has been carried out (search also IF the modification was refused…)
   - Expected Behaviour (1): the modification request is refused
   - Expected Behaviour (2): the search reveals no change in the tester's profile

4. Illegal Registrations [If tester registered]
   - Tests: if the remote DF overwrites existing registrations when new ones are made
   - Actions (hacker): try to register tester agent again, search to confirm no changes were made
   - Expected Behaviour (1): refusal – either due to an "already registered" error or due to the fact that hacker cannot register "tester"
   - Expected Behaviour (2): no changes made in testers profile

5. Illegal Deregistration [if tester registered]
   - Tests: if the remote DF allows an agent to deregister another agent
   - Actions (hacker): try to deregister tester agent again, search to confirm no changes were made
   - Expected Behaviour (1): refusal of request
   - Expected Behaviour (2): no changes made in testers profile

6. Deregister [if tester registered]
   - Action (tester): deregister.

3.7.3 IV.iii: DF Federation

(Note that in this sequence it is assumed that the tester agent can also act as a DF, although this may be another agent.)

1. DF Register
   - Tests: if a remote DF can register itself with the target platforms DF.
   - Action (tester): the tester registers itself as a DF with the DF on the remote platform. (and registers a dummy service "df_federation_test" with itself)
   - Expected Behaviour: request protocol adhered to and registration accepted.

2. Non-Federated search [If DF registered]
   - Tests: that normal (unfederated) is still possible.
   - Action (tester): Send a search request for the service "df_federation_test" to the target platform DF with max-depth set 0 (no federated search)
   - Expected Behaviour: service not found (failure)

3. Federated search [If DF registered]
   - Tests: if the remote DF supports federated search.
   - Action (tester): Send a similar request to the above but using max search depth 1.
   - Expected Behaviour (1): The remote DF queries the tester agent about the named service with max search depth decreased by 1.
   - Expected Behaviour (2): The service is found (at *least* once) and the result returned to the tester agent.

4. Deregister [If DF registered]

References

[Agentcities] Agentcities Project Homepage http://www.agentcities.org/

[Camenen00] “Solving Interoperability Issues for an Open Agent Service Architecture”, Y. Camenen

[Charlton et. al. 00] “Dealing with Interoperability for Agent Based Services”, P. Charlton, D. Bonnefoy, N. Lhuillier, A. Gouaich, Y. Camenen. October 2000, On-line at http://leap.crm-paris.com/agentcities.

[FIPA] Foundation for Intelligent Physical Agents Specifications can be found at http://www.fipa.org/