



Inter-platform communication

Step by step guide

The AgentService Team

Copyright @ 2007 – l.i.d.o. – DIST University of Genoa

Abstract

This is a very brief guide with the only aim to make users ready to develop agents a distributed environment. These agents are now able to interact among them in a simple way, by using the *.NET Remoting* technology.

Index of Contents

Abstract	2
Index of Contents.....	3
1. Introduction	4
2. Configuration	4
3. Using remoting to exchange messages	6

1. Introduction

AgentService agents are able to speak with peers on the same platform with the simple message exchange or through conversations.

These solutions represent a good way for the communication among agents and then the same conception is exported to the *inter-platform* message exchange.

Basically, from the agent point of view, to send a message to a peer on the same platform or on a federated one is exactly the same thing. As you can see in next chapter, only few parameters must be configured.

2. Configuration

In order to setup the remoting server on your AgentService you have to configure three configuration files. The first file you must configure is *install.xml*. If you want to manage a distributed environment with several AgentService platform, every platform must have its *name*, different from the other ones. So type a unique name for every installation, setting the *name* attribute in the description tag (*AgentService* is the default name).

```
<description name="AgentService" mobile="false" dynamic="true"
transport-profile="[none]" />
```

It is better to not change the name of a yet installed platform, but to install a new platform with the new name.

Modified the name of the platform, you can now install the platform and then configure the other two files.

Open the *msmq.conf* in the `\modules\core\conf` directory. It is the configuration file for the messaging module. Regarding the remoting service, you are interested in the following parameters:

- *FederationConfig*: the path of the file containing the list of the federated platforms.
- *EnableRemoteSend*: true if you want to enable the remoting service to send messages.
- *EnableRemoteReceive*: true if you want to enable the remoting service to receive messages.

- *RemotingPort*: the port where the remoting service is listening.

The file containing the list of federated platforms is usually contained in the same directory of *msmq.conf* (*\modules\core\conf*) but you can put it everywhere. Obviously set *EnableRemoteSend* and *EnableRemoteReceive* to true (it is better to write *True*).

Finally, set the *RemotingPort* parameter. Remember that if you are testing the remoting service among two platforms on the same PC, you must set a different port for each platform.

Here you can find a simple example of *msmq.conf*:

```
# DumpInterval: the number of maximum changes on an agent message queue
between two queue serializaions
#
DumpInterval 100

# DumpPath: the path to the serialization directory
#
DumpPath C:\ThePlatformForRemoting\messaging

# EnableEvents: boolean flag that is used to control the event firing
for this module
#           if true, events are risen at each operation on the
module
#
EnableEvents False

# VerboseMode: boolean flag that is used to control which events are
fired. If true all
#           the events are fired and also those which can occur
frequently (GetAll, Peek, and Query verbs)
#
VerboseMode False

# FederationConfig: the file containing the federated platforms
#
FederationConfig C:\ThePlatformForRemoting\modules\core
                \conf\federation.conf

# EnableRemoteSend: boolean flag that is used to activate the feature
of sending messages to other platforms.
#
EnableRemoteSend True

# EnableRemoteReceive: boolean flag that is used to activate the
feature of sending messages to other platforms.
#
EnableRemoteReceive True

# RemotingPort: the port where the remoting server is listening
#
RemotingPort 5000
```

Regarding the list of the federated platforms, see the following example:

```
<?xml version="1.0" encoding="utf-8"?>
<federation xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <platform name="WinterMute" ip="130.251.22.12" port="60009" />
  <platform name="Continuity" ip="130.251.22.103" port="5000"/>
</federation>
```

In this example (the file is *platform.conf*) two federated platforms are listed: *WinterMute* and *Continuity*. They have an IP address and a port (the port you set in the *msmq.conf* of the respective platforms).

You can modify the file every time you want, also during the platform execution and the modification is immediately taken into account.

3. Using remoting to exchange messages

Send a message to a remote agent is pretty easy, if you followed the configuration steps. You can continue to use both the simple message exchange and the conversation objects. The only different thing is that you must specify in the agent AID of the recipient agent the name of the remote platform.

So, if you use the simple message exchange:

```
AgentMessage msg = new AgentMessage();
msg.Envelope.To.Add(new AID("Marly", "Continuity"));
msg.Body.Add("content", "Greetings");
this.Runtime.MessageSvc.SendMessage(msg);
```

The sender agent (for instance, named *Virek*) sends a message to *Marly* and it must to specify the platform where *Marly* is running, that is *Continuity*.

Regarding the conversations, as you see, it is the same:

```
IConversation conv = this.Runtime.MessageSvc.OpenConversation(new
AID("Virek", "WinterMute"));
```

The receiver agents receive messages and manage incoming conversations without caring if the sender is running on a remote platform.