Quick Start Guide

Running the AgentService Framework in 15 minutes

The AgentService Team

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

Abstract

This tutorial will help you in quickly installing, running and playing the Demo with the AgentService framework. The purpose of this tutorial is not to provide a deep understanding of the details of the framework but is to just get you acquainted with the content of the source tree of the framework and to provide you with all the basics information required to let the framework installed and working.

Even though some general information about the structure of the system will be given, this is not a *programming guide*. For this reason, you will not find here a detailed discussion about how to build software agents with AgentService and take full advantage of the features of the framework.

Index of Contents

1. Introduction 4
2. Installing AgentService 5
2.1. Getting the Software 5
2.2. Inspecting the libraries of the distribution 6
2.3. The AgentDemo project 6
2.4. Compiling the Software 8
2.5. Installing the Framework 8
3. Running the AgentService Platform 10
3.1. Platform First Execution 10
3.2. Stopping the Platform 11
3.3. Subsequent Executions of the Platform 12
3.4. Platform Configuration File 12
4. Running the Auction Demo 13
4.1. The AgentDemo Application 13
4.2. Platform Batch Execution 15
4.3. Running the AuctionBatch 16
5. Things to Know 17

1. Introduction

AgentService is a framework for developing agent oriented applications. It supports developers in implementing agent-based applications by providing them with:

- a library and full set of APIs for creating software agents;
- a run-time environment, which is the AgentService platform, on which agent-based applications run;
- a set of tools supporting the development and the management of agent-based applications.

Agent-based applications are software applications which are composed by software agents. AgentService proposes its own model of software agent which is generally enough to implement the different agent models devised by the community of researchers and that is based on the concept of ***knowledge objects*** which define the state of the agent, and ***behavior objects*** that define the aggregate activity of agents. A collection of interoperating software agents defines a multi-agent system which is the outcome of agent-base development. AgentService allows implementing multi-agent systems where software agents are defined and implemented according to the model proposed by the framework.

AgentService is based on the Common Language Infrastructure and needs the ***Microsoft .NET Framework 2.0*** or the ***Mono 1.2.3.1*** or higher distribution in order to run. Hence, in order to install and use it, it is necessary to have one of these frameworks installed.

This quick start guide refers to the pre-release version of AgentService (codename ***Macumba***), which has been delivered to the whole community with a subset of the entire features of the framework. This version already allows you develop agent-based applications, but does not comprise the following feauters:

- integrated, GUI based installer for the Linux and the Windows operating systems;
- support for mobility and distributed multi-agent systems;
- integrated GUI for the visual monitoring and management of the platform;

Despite the lack of these features, this version of the framework allows you to easily program multi-agent system. The previously cited features will be delivered soon and are now under test.

2. Installing AgentService

2.1. Getting the Software

This quickstart guide is included into the pre-release distribution of AgentService, which can be freely downloaded at the following address: http://www.agentservice.it/download/agentservice-1.0-pre-release.zip. The internal structure of the zip archive is depicted in *Figure 1*.

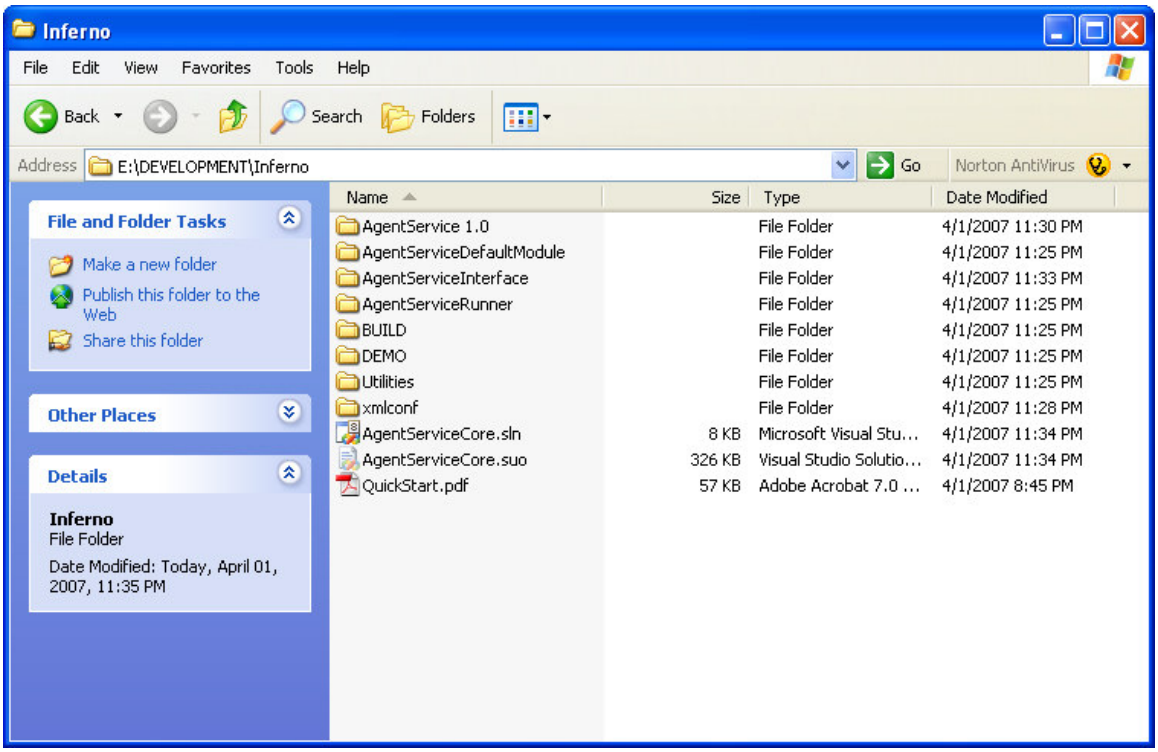

Figure 1. The file system of the distribution.

The zip file contains many folders. We can distinguish a set of folders which constitute core of the AgentService framework. These folders identify the projects which are managed by the ***AgentServiceCore.sln*** Visual Studio 2005 Solution:

- ***AgentService 1.0***: it contains the components that constitute the core of the system. This library defines the software platform, the agent model and all the internal components required by the framework to run. This project is compiled into the ***AgentService.dll*** library.

- ***AgentServiceDefaultModule***: contains the implementation of the AgentService core modules. These modules are required by the AgentService platform to set up a platform instance and constitute also a reference implementation for developers that want to add functionalities to the system through modules. By customizing the installation of the platform it is possible to install and use other modules than these ones, which are provided by the framework. This project is compiled into the ***AgentServiceDefaultModules.dll*** library.

- ***AgentServiceInterface***: contains the interface and the basic types which are required to interface client software with a running instance of the AgentService platform. This project is compiled into the ***AgentServiceInterface.dll***.

- ***AgentServiceRunner***: contains the platform driver. This project is compiled into the executable ***asdrv.exe*** which is required to star the platform through the console.

- ***Utilities:*** contains the definition of some utility classes which are used by the framework, that are compiled into the ***Utilities.dll***.

- ***xmlconf:*** contains the classes which are required to read and write xml configuration and installation files which are compiled into the library ***xmlconf.dll***.

Once we opened the zip archive in order to continue it is necessary to deflate it into a folder of the local system. The selection of the target folder is not important for the installation process since the installation path will be selected through the xml installation file.

2.2. Inspecting the libraries of the distribution.

A precompiled version of the libraries and of the platform driver can be found into the ***BUILD*** folder along with their debug symbols. Into the same folder you can find the default xml installation file ***install.xml***. The content of this folder is displayed into *Figure 2*.

2.3. The AgentDemo project

The ***DEMO*** folder contains the ***AgentDemo*** project which is a simple implementation of an agent-based auction and that will be discussed in Section 4, and it is not directly managed by the ***AgentServiceCore.sln*** solution file. The content of the ***DEMO*** folder is depicted in *Figure 3*.

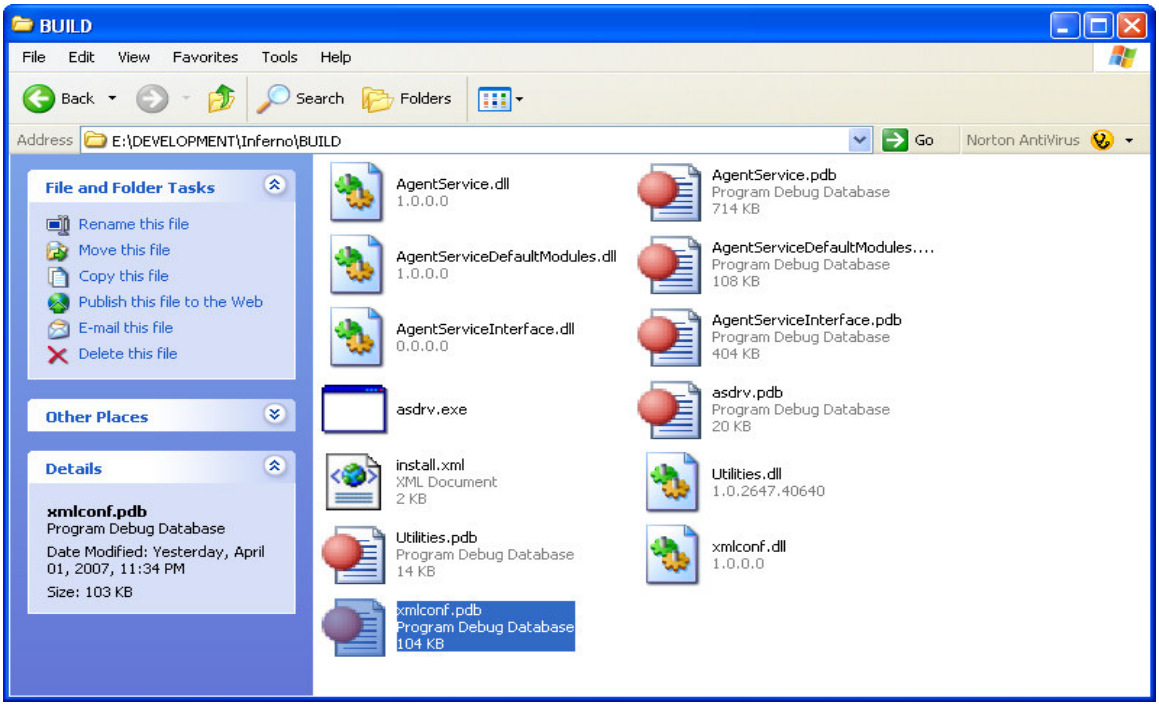

Figure 2. Content of the BUILD folder.

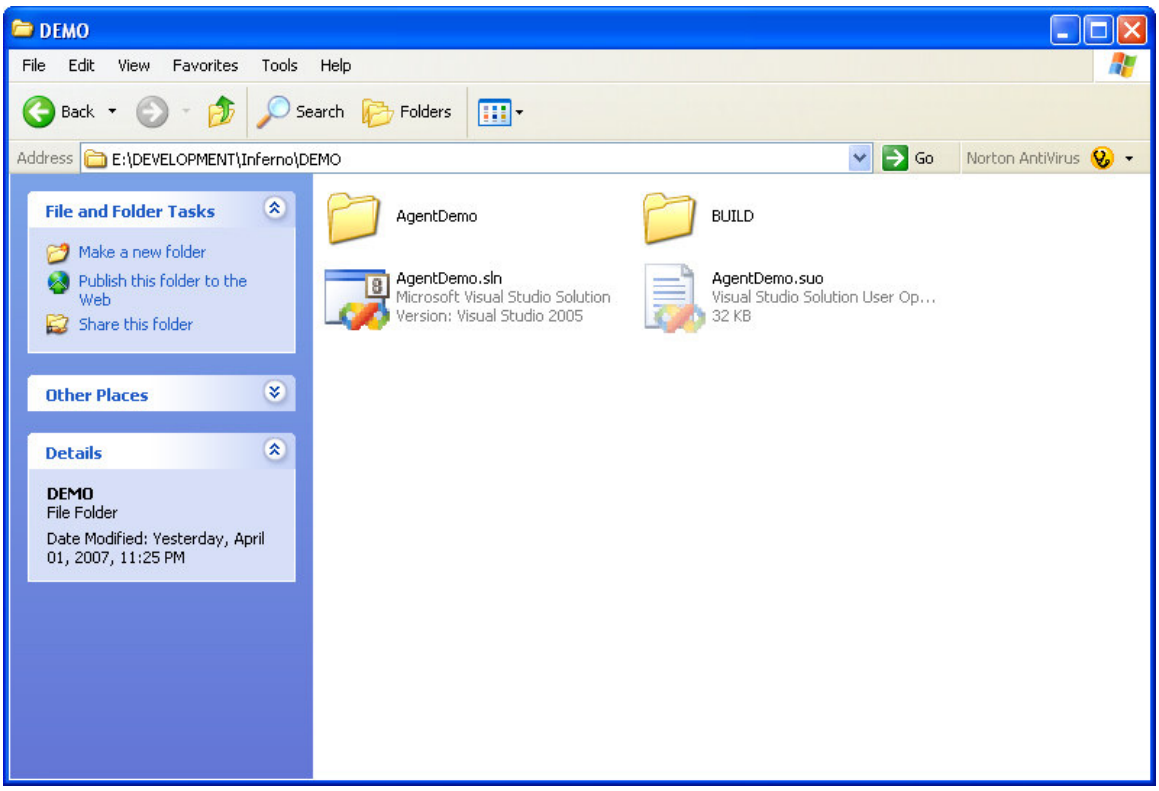

Figure 3. Content of the DEMO folder.

2.4. Compiling the Software

If you want to recompile the entire software you actually need the Visual Studio.Net 2005 development environment. In order to build the core of the framework you need to open the ***AgentServiceCore.sln*** solution file and Rebuild the entire solution. The demo project can be built by opening the ***AgentDemo.sln*** located in the ***DEMO*** folder.

The output of the build process is copied by the Visual Studio build engine into the corresponding ***BUILD*** directories.

2.5. Installing the Framework

The ***BUILD*** folder depicted in *Figure 1* contains all the executable and component libraries required to install the AgentService framework. In order to install the framework it is necessary to run the ***asdrv.exe*** platform driver with the install switch and by specifying the installation file. This pre-release distribution comes with a default installation file which is ***install.xml***. The content of this file is depicted into *Figure 4*.

```
  <?xml version="1.0" encoding="utf-8" ?>
- <installation>
    <base-dir path="c:\Program Files\AgentService\" />
    <repository file="install.dat" />
    <description name="AgentService" mobile="false" dynamic="true" transport-profile="[none]" />
  - <platform>
      <factory persistence="true" full-type-name="AgentService.Agent.LightAgentFactory" assembly-file="AgentService.dll" />
      <policy full-type-name="AgentService.Platform.NoDomainPolicy" assembly-file="AgentService.dll" />
      <remote enable="false" port="9000" secure="false" />
    </platform>
  - <core>
      <storage full-type-name="AgentService.Platform.Modules.Default.DefaultStorageModule" assembly-
        name="AgentServiceDefaultModules.dll" config-file="storage.conf" />
      <persistence full-type-name="AgentService.Platform.Modules.Default.DefaultPersistenceModule" assembly-
        name="AgentServiceDefaultModules.dll" config-file="persistence.conf" />
      <messaging full-type-name="AgentService.Platform.Modules.Default.DefaultMessagingModule" assembly-
        name="AgentServiceDefaultModules.dll" config-file="messaging.conf" />
      <logging full-type-name="AgentService.Platform.Modules.Default.DefaultLoggingModule" assembly-
        name="AgentServiceDefaultModules.dll" config-file="logging.conf" />
    </core>
    <additional />
    <properties />
  </installation>
```

Figure 4. Content of the install.xml file.

The default settings of the installation file instruct the platform driver to install the platform into the ***C:\Program Files\AgentService*** folder and configures it with the default core modules contained into the ***AgentServiceDefaultModules.dll*** library.

It is possible to change the installation path of the platform by providing a different directory into the *path* attribute of the *base-dir* node of the XML document. Other elements that can be easily changed are the description of the platform which just provides textual information of the installed platform.

Note for Linux/MacOS users: the installation path is the one that **must** be necessarily changed in order to successfully install the platform. This pre-release distribution automatically adopts the *Windows* convention for generating the paths. Hence, when installing the platform on *Linux* (or *BSD*) based operating system the value of this path needs to be changed.

The set of core modules that are installed are controlled by the *core* node of the XML document. As depicted in *Figure 4* the default installation settings install the default core modules. These modules do not require any installation file and the configuration file specified are those that will be used when the system is running. Additional modules to install can be added to file by adding *<module full-type-name="..." assembly-name="..." config-file="..." />* to the *additional* node.

Additional and advanced properties can be controlled through the installation file like:

- the properties of the remoting service (*remote* node);
- the selection of some internal components like the agent factory and the factory policy adopted to control the creation of software agents (*factory* and *policy* nodes);
- additional properties in the form of *<property name="n" value="v" />* contained into the *properties* node[1];

If the default settings are acceptable it is possible to perform the installation process. In order to install the platform it is necessary to:

- open a console (that is the Command Prompt on Windows systems);
- locate the and move to the ***BUILD*** directory;
- run the following command:
  - ***asdrv.exe –i:install.xml*** (for the Microsoft .NET runtime);
  - ***mono asdrv.exe –i:install.xml*** (for the Mono runtime);
- wait for the execution of the installer to terminate.

[1] The properties node is a useful place to insert additional configuration values that can need to be accessed by some custom internal platform components and makes either the platform installation file or the platform configuration file an open configuration system.

The outcome of this action is depicted in *Figure 5*, which displays the messages that the installer writes to the console while performing the installation process. If the installation process is successful all the operations ends with the ***[OK]*** message and the ***Installation successful*** terminates the console log.

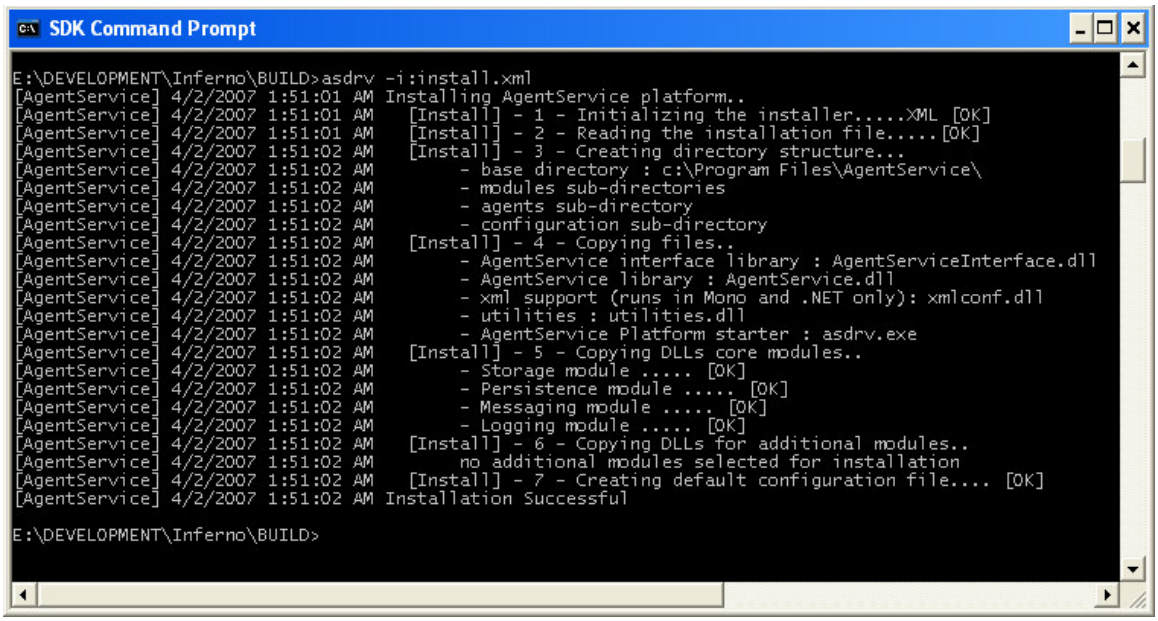

Figure 5. Installation log on a Windows system.

The installation of the platform is a two phase process. The first step constitutes in launching the platform driver with the installation switch as previously described. The second step terminates the process when the platform runs for the first time. This operation will be described in the next section.

3. Running the AgentService Platform

3.1. Platform First Execution

In order to run the platform you need to change your current directory to the AgentService platform installation directory (if not changed is C:\Program Files\AgentService).

The first activation of the platform completes the installation process. This is done by executing the following command:

- ***asdrv.exe*** (on Windows systems);
- ***mono asdrv.exe*** (on Mono systems);

During the first execution of the platform the modules complete their installation. Before activating the platform the driver looks at the platform configuration file and for each module executes the pending installations. When this process finishes the modules are loaded, the core agents activated, and platform moves into the ***Running*** state. When the platform is running clients can connect and interact with the platform to create software agents.

```
SDK Command Prompt - asdrv

E:\DEVELOPMENT\Inferno\BUILD>c:

C:\Program Files>cd AgentService

C:\Program Files\AgentService>asdrv
[AgentService] 4/2/2007 2:22:46 AM Platform start-up
[AgentService] 4/2/2007 2:22:46 AM   [Binding] - 1 - Reading configuration file..... [OK]
[AgentService] 4/2/2007 2:22:47 AM   [Binding] - 2 - Creating installation information repository..... [OK]
[AgentService] 4/2/2007 2:22:47 AM   [Binding] - 3 - Installing Core Modules...
[AgentService] 4/2/2007 2:22:47 AM       - Storage..... [OK]
[AgentService] 4/2/2007 2:22:47 AM          type : AgentService.Platform.Modules.Default.DefaultStorageModule
[AgentService] 4/2/2007 2:22:47 AM          assembly : AgentServiceDefaultModules.dll
[AgentService] 4/2/2007 2:22:47 AM          config file: storage.conf
[AgentService] 4/2/2007 2:22:47 AM       - Persistence..... [OK]
[AgentService] 4/2/2007 2:22:47 AM          type : AgentService.Platform.Modules.Default.DefaultPersistenceModule
[AgentService] 4/2/2007 2:22:47 AM          assembly : AgentServiceDefaultModules.dll
[AgentService] 4/2/2007 2:22:47 AM          config file: persistence.conf
[AgentService] 4/2/2007 2:22:47 AM       - Messaging..... [OK]
[AgentService] 4/2/2007 2:22:47 AM          type : AgentService.Platform.Modules.Default.DefaultMessagingModule
[AgentService] 4/2/2007 2:22:47 AM          assembly : AgentServiceDefaultModules.dll
[AgentService] 4/2/2007 2:22:47 AM          config file: msmq.conf
[AgentService] 4/2/2007 2:22:47 AM       - Logging..... [OK]
[AgentService] 4/2/2007 2:22:47 AM          type : AgentService.Platform.Modules.Default.DefaultLoggingModule
[AgentService] 4/2/2007 2:22:47 AM          assembly : AgentServiceDefaultModules.dll
[AgentService] 4/2/2007 2:22:47 AM          config file: logging.conf
[AgentService] 4/2/2007 2:22:47 AM   [Binding] - 4 - Installing Additional Modules...
[AgentService] 4/2/2007 2:22:47 AM     no additional modules selected for installation
[AgentService] 4/2/2007 2:22:47 AM Platform bound [Idle mode]
[AgentService] 4/2/2007 2:22:47 AM Platform Starts...
[AgentService] 4/2/2007 2:22:47 AM      [Starting] - 1 - Loading Core Modules...
[AgentService] 4/2/2007 2:22:47 AM              - Storage Module loading..... Default Storage Module [OK]
[AgentService] 4/2/2007 2:22:47 AM              - Messaging Module loading..... Default Messaging Module [OK]
[AgentService] 4/2/2007 2:22:47 AM              - Persistence Module loading..... Default Persistence Module [OK]
[AgentService] 4/2/2007 2:22:47 AM              - Logging Module loading..... Default Logging Module [OK]
[AgentService] 4/2/2007 2:22:47 AM      [Starting] - 2 - Loading Additional Modules...
[AgentService] 4/2/2007 2:22:47 AM              - No Additional Modules to Load
[AgentService] 4/2/2007 2:22:47 AM Platform ready.
[AgentService] 4/2/2007 2:22:47 AM Platform activating...
[AgentService] 4/2/2007 2:22:47 AM      [Activating] - 1 - Activating Factory...
[AgentService] 4/2/2007 2:22:47 AM              - factory assembly: AgentService.dll
[AgentService] 4/2/2007 2:22:47 AM              - factory type: AgentService.Agent.LightAgentFactory
[AgentService] 4/2/2007 2:22:47 AM              - policy assembly: AgentService.dll
[AgentService] 4/2/2007 2:22:47 AM              - policy type: AgentService.Platform.NoDomainPolicy
[AgentService] 4/2/2007 2:22:47 AM      [Activating] - 2 - Activating Core Agents...
[AgentService] 4/2/2007 2:22:47 AM              - activating AMS (Agent Management System)..... [OK]
[AgentService] 4/2/2007 2:22:47 AM              - activating MTS (Message Transport System)..... [OK]
[AgentService] 4/2/2007 2:22:47 AM              - activating DF (Directory Facilitator)..... [OK]
[AgentService] 4/2/2007 2:22:47 AM      [Activating] - 3 - Activating User Agents...
[AgentService] 4/2/2007 2:22:47 AM Platform Running
```

Figure 6. First execution of the AgentService Platform.

3.2. Stopping the Platform

It is possible to stop the activity of the platform by pressing ***Enter*** in the console. The platform stops its execution, the service agents (MTS, DF, and AMS) are stopped, and modules unloaded.

```
SDK Command Prompt
[AgentService] 4/2/2007 2:34:36 AM Platform is stopping..
[AgentService] 4/2/2007 2:34:36 AM      [Stopping] - 2 - Stopping and Persisting User Agents...
[AgentService] 4/2/2007 2:34:36 AM      [Stopping] - 1 - Stopping and Persisting Core Agents...
[AgentService] 4/2/2007 2:34:36 AM              - stopping and persisting ams ..... [OK]
[AgentService] 4/2/2007 2:34:36 AM              - unloading any remaining AppDomain...
[AgentService] 4/2/2007 2:34:36 AM                ... all AppDomain unloaded [OK]
[AgentService] 4/2/2007 2:34:36 AM Platform Stopped [ready mode].
[AgentService] 4/2/2007 2:34:36 AM Platform is Shutting Down...
[AgentService] 4/2/2007 2:34:36 AM      [Shutting] - 3 - Unloading Additional Modules...
[AgentService] 4/2/2007 2:34:36 AM              - No Additional Modules to Unload
[AgentService] 4/2/2007 2:34:36 AM      [Shutting] - 2 - Unloading Core Modules...
[AgentService] 4/2/2007 2:34:36 AM              - Logging Module [Default Logging Module] ...... [OK]
[AgentService] 4/2/2007 2:34:36 AM              - Persistence Module [Default Persistence Module] ...... [OK]
[AgentService] 4/2/2007 2:34:36 AM              - Messaging Module [Default Messaging Module] ...... [OK]
[AgentService] 4/2/2007 2:34:36 AM              - Storage Module [Default Storage Module] ...... [OK]
[AgentService] 4/2/2007 2:34:36 AM      [Shutting] - 1 - Writing to Configuration File..... [OK]
[AgentService] 4/2/2007 2:34:36 AM Platform shut down.

C:\Program Files\AgentService>
```

Figure 7. Stopping the AgentService Platform.

```
SDK Command Prompt - asdrv
[AgentService] 4/2/2007 2:34:36 AM      [Shutting] - 1 - Writing to Configuration File..... [OK]
[AgentService] 4/2/2007 2:34:36 AM Platform shut down.

C:\Program Files\AgentService>asdrv
[AgentService] 4/2/2007 2:43:06 AM Platform start-up
[AgentService] 4/2/2007 2:43:06 AM   [Binding] - 1 - Reading configuration file..... [OK]
[AgentService] 4/2/2007 2:43:07 AM   [Binding] - 2 - Binding installation information repository..... [OK]
[AgentService] 4/2/2007 2:43:07 AM Platform bound [Idle mode]
[AgentService] 4/2/2007 2:43:07 AM Platform Starts...
[AgentService] 4/2/2007 2:43:07 AM      [Starting] - 1 - Loading Core Modules...
[AgentService] 4/2/2007 2:43:07 AM              - Storage Module loading..... Default Storage Module [OK]
[AgentService] 4/2/2007 2:43:07 AM              - Messaging Module loading..... Default Messaging Module [OK]
[AgentService] 4/2/2007 2:43:07 AM              - Persistence Module loading..... Default Persistence Module [OK]
[AgentService] 4/2/2007 2:43:07 AM              - Logging Module loading..... Default Logging Module [OK]
[AgentService] 4/2/2007 2:43:07 AM      [Starting] - 2 - Loading Additional Modules...
[AgentService] 4/2/2007 2:43:07 AM              - No Additional Modules to Load
[AgentService] 4/2/2007 2:43:07 AM Platform ready.
[AgentService] 4/2/2007 2:43:07 AM Platform activating...
[AgentService] 4/2/2007 2:43:07 AM      [Activating] - 1 - Activating Factory...
[AgentService] 4/2/2007 2:43:07 AM              - factory assembly: AgentService.dll
[AgentService] 4/2/2007 2:43:07 AM              - factory type: AgentService.Agent.LightAgentFactory
[AgentService] 4/2/2007 2:43:07 AM              - policy assembly: AgentService.dll
[AgentService] 4/2/2007 2:43:07 AM              - policy type: AgentService.Platform.NoDomainPolicy
[AgentService] 4/2/2007 2:43:07 AM      [Activating] - 2 - Activating Core Agents...
[AgentService] 4/2/2007 2:43:07 AM              - activating AMS (Agent Management System)..... [OK]
[AgentService] 4/2/2007 2:43:07 AM              - activating MTS (Message Transport System)..... [OK]
[AgentService] 4/2/2007 2:43:07 AM              - activating DF (Directory Facilitator)..... [OK]
[AgentService] 4/2/2007 2:43:07 AM      [Activating] - 3 - Activating User Agents...
[AgentService] 4/2/2007 2:43:07 AM Platform Running
```

Figure 8. Second and subsequent executions of the Platform.

3.3. Subsequent Executions of the Platform

If we run again the AgentService platform we will not notice the ***[Binding] - 3*** stage that is depicted in *Figure 6*. This is the difference between the first run and the subsequent executions of the AgentService platform. Another difference is that if the platform was previously executing software agents that did not terminate their job these agents – if persistent – are automatically restarted and resume from the persistence system. This is what happens to the AMS, DF, and MTS agents which are persistent.

3.4. Platform Configuration File

The platform controls its start-up through a configuration file. By default this file is named ***platform.conf*** and is saved into the ***conf*** sub-directory of the platform installation folder. The content of this file is displayed in *Figure 9*. As we can see the content of the file is more or less the same of the installation

file with a slightly different layout. In particular some information contained in specific nodes of the installation file has been saved in the properties node.

It is interesting to notice that the *platform* node exposes an attribute which is called *first-start* that informs the driver if the current execution is the first of the platform. Setting this attribute to ***True*** can be useful when we need to run the first execution again.

```
  <?xml version="1.0" encoding="utf-8" ?>
- <platform first-start="False" skip-failed-modules="False">
    <base-dir path="C:\Program Files\AgentService\" />
    <repository file="install.dat" />
    <description name="AgentService" mobile="false" dynamic="true" transport-profile="[none]" />
  - <modules>
    - <core bin-path="modules\core\bin\" conf-path="modules\core\conf\">
        <storage key="1" full-type-name="AgentService.Platform.Modules.Default.DefaultStorageModule" assembly-
          name="AgentServiceDefaultModules.dll" config-file="storage.conf" />
        <persistence key="2" full-type-name="AgentService.Platform.Modules.Default.DefaultPersistenceModule" assembly-
          name="AgentServiceDefaultModules.dll" config-file="persistence.conf" />
        <messaging key="3" full-type-name="AgentService.Platform.Modules.Default.DefaultMessagingModule" assembly-
          name="AgentServiceDefaultModules.dll" config-file="msmq.conf" />
        <logging key="4" full-type-name="AgentService.Platform.Modules.Default.DefaultLoggingModule" assembly-
          name="AgentServiceDefaultModules.dll" config-file="logging.conf" />
      </core>
      <additional />
    </modules>
  - <properties>
      <property name="system.factory.typename" value="AgentService.Agent.LightAgentFactory" />
      <property name="system.remote.enable" value="False" />
      <property name="system.factory.assembly" value="AgentService.dll" />
      <property name="system.policy.typename" value="AgentService.Platform.NoDomainPolicy" />
      <property name="system.remote.secure" value="False" />
      <property name="system.persistagent" value="True" />
      <property name="system.remote.tcpport" value="9000" />
      <property name="system.policy.assembly" value="AgentService.dll" />
    </properties>
  </platform>
```

Figure 9. Platform configuration file.

The driver also gives the possibility of running the platform with a different platform configuration file through the switch ***-r:path-to-the-new-file***.

4. Running the Auction Demo

4.1. The AgentDemo Application

The AgentDemo application is a multi-agent system which implements an auction system and uses as infrastructure the AgentService platform. The project creates a library – ***AgentDemo.dll*** - which contains all the code required to set up and run the multi-agent system with AgentService.

The system consists of two different types of agents which interoperate to create an auction:

- the ***Auctioneer*** agent: this kind of agent represents the auctioneer. It opens the auctions, publishes the items, and waits for the bidders. When the bidders register themselves to participate in the auction the ***Auctioneer*** agent manages all the offers and then establishes t he winner;

- the ***Bidder*** agent: bidder agents represent people which participate in the auction, they have a budget and some preferences that drive them in deciding if participate in a given auction or not and determinate the amount of money they are allowed to offer.

The interaction between the bidders and the auctioneer is the following:

- the auctioneer open the auction for the registration and publishes the item which is sold;

- the bidders join the auction (according to their preferences);

- the auctioneer closes the registration, and starts auction with a base price;

- bidders are asked to make an offer and the auctioneer sends back to the bidders which is the bidder currently winning;

- bidders can decide to make another offer or retire;

- the process terminates when there is only one bidder which becomes the winner.

The demo is configured to run an auction with one auctioneer and four bidders. These settings can be changed by editing the configuration file of the demo.

More details of the structure of the agents and the auction process can be found in the ***README.[ITA|EN].txt*** that are located in the ***AgentDemo*** folder contained in the ***DEMO*** directory.

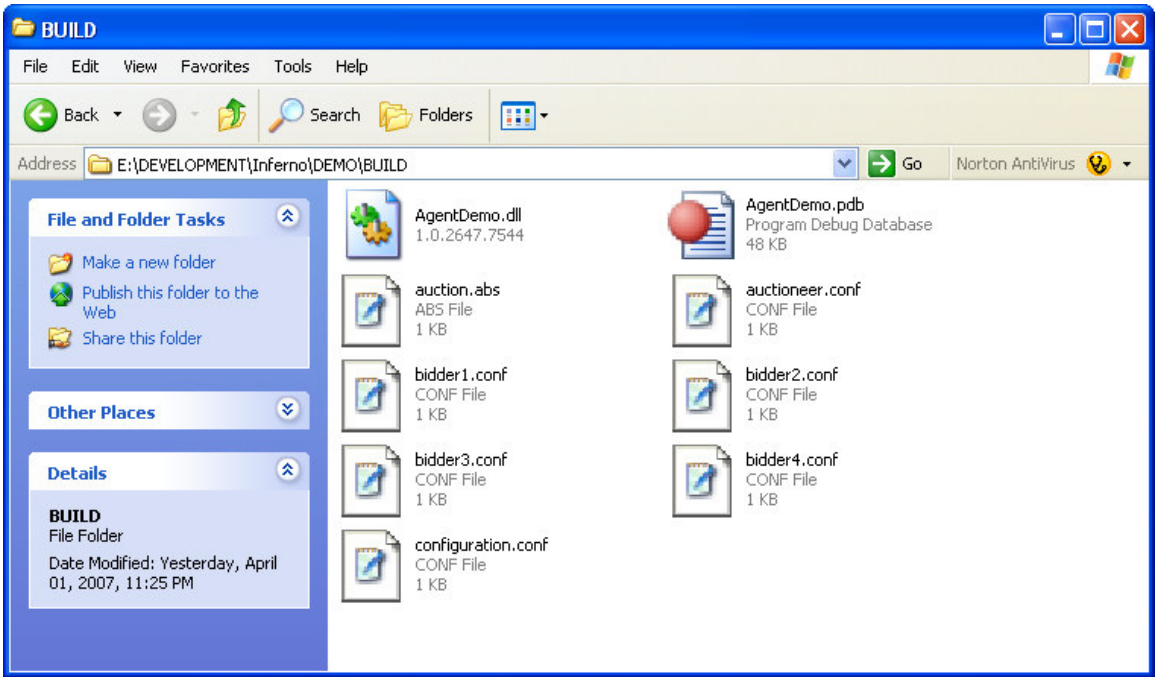

Figure 10. Files Required to Run the AgentDemo Application.

In order to execute this demo it is necessary to copy the content of the ***DEMO\BUILD*** folder into the base installation directory of the installation platform and run the platform in batch mode.

This directory contains the ***AgentDemo.dll*** library and its debug symbols, along with all the configuration files required to run the multi-agent system. In particular the configuration files are:

- ***auction.abs***: file required to run the platform in batch mode;
- ***configuration.conf***: configuration file of the batch which contains the number and the type of agents to create;
- ***auctioneer.conf***: configuration file of the auctioneer agent;
- ***bidder[1-4].conf***: configuration files of the bidder agents.

It is possible to change the number of agents constituting the multi-agent system by editing the ***configuration.conf*** file and providing the related configuration files.

4.2. Platform Batch Execution

The platform can be programmatically instructed to run a multi-agent system through the use of batches. In order to execute a batch it is necessary to run

the platform driver with execute batch switch (***-x:***) followed by the name of the batch file:

- ***asdrv.exe -x:batch_file.abs*** (on Windows systems);
- ***mono asdrv.exe -x:batch_file.abs*** (on Mono systems);

When the driver is executed in batch mode, it starts the platform, and then passes the control of the platform instance to a batch program dynamically loaded according to the information stored in the batch file. The batch program receives an interface to the platform and can issue the same commands and client software applications. Through the batch we can change the state of the platform, create agents, activate the multi-agent system, and stop the platform.

Batch programs are specified into batch files *(**.abs** - AgentService **B**atch **S**cript)* which always have the following structure:

```
file=[path-to-batch-configuration-file]
conf=[relative-path-to-platform-configuration-file]
type=[full-type-name-of-batch-program]
assembly=[path-to-the-assembly-of-the-batch-program]
```

The first directive, which is optional, specifies the configuration file required by the batch program to run, if it is necessary. The second directive identifies the relative path to the platform configuration file that should be used by the driver. The third directive specifies the full type name of the batch program that will be dynamically loaded and executed by the driver. The fourth directive identifies the path to the assembly where the batch program is defined.

4.3. Running the AuctionBatch

The ***AgentDemo*** application is set up and activated by executing the batch program described identified in the ***auction.abs*** file. In the case of the ***AgentDemo*** this file is configured as follows:

```
file=configuration.conf
conf=conf\platform.conf
type=AgentDemo.AuctionBatch
assembly=AgentDemo.dll
```

Once we have copied all the files depicted in *Figure 10*, we run the system by issuing the following command in the platform base directory:

- ***asdrv.exe -x:auction.abs*** (on Windows systems);

- ***mono asdrv.exe -x:auction.abs*** (on Mono systems);

The driver will then activate the platform, load the AgentDemo.dll module in the platform, create the five agents constituting the multi-agent system (MAS), activate the MAS, and wait for the end of the auction. A detailed log of the activities of the auction can be seen in *Figure 11*, which captures a snapshot of the execution of the multi-agent system.

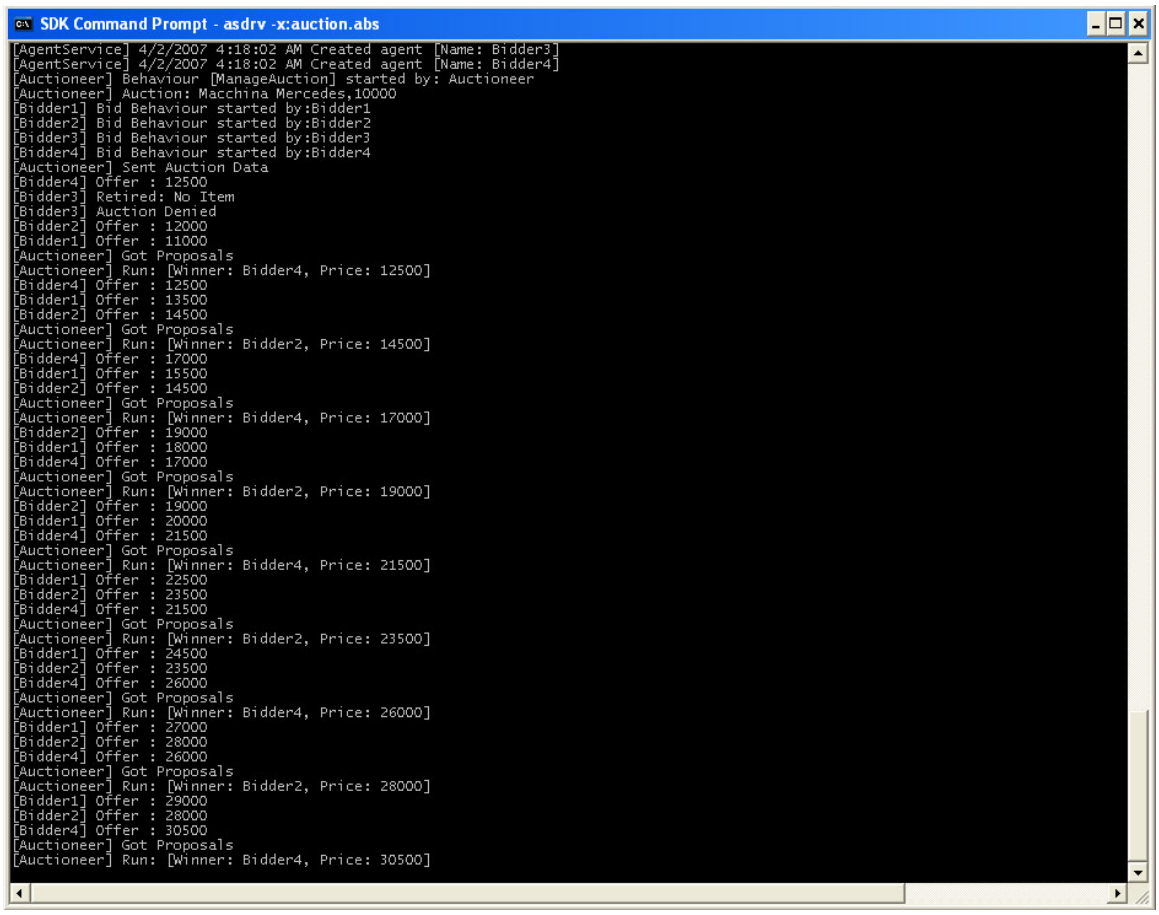

Figure 11. Execution of the AgentDemo Batch.

5. Things to Know

The pre-release version of AgentService has been delivered to the public in order to give an idea of its basic features and it is still missing of some features that are under test. In particular the AgentService development team is working on the following issues:

- testing the features of the GUI-based interface to AgentService;

- implementing the remote access to the platform;
- solving some execution problems with the Mono runtime;
- testing the coordination of distributed AgentService platform instance on version 2.0 of the Common Language Infrastructure.

For all these reasons some of the features which can be configured to the installation and configuration files are ignored (basically the remote service).

For what concerns the execution on the Mono runtime the platform can be installed and run with Mono but at present time there are some bugs in the creation of the agents due to a different implementation of the Reflection APIs.