

Grasshopper User's Guide

Release 2.2

IKV++ GmbH
Bernburger Strasse 24-25
10963 Berlin, Germany
<http://www.grasshopper.de>

Copyright (c) 2000 IKV++ GmbH Informations- und Kommunikationssysteme

All Rights Reserved.

Grasshopper Release 2.2 User's Guide, März 2001

The *Grasshopper User's Guide* is copyrighted and all rights are reserved. Information in this document is subject to change without notice and does not represent a commitment on the part of IKV++ GmbH. The document may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from IKV++ GmbH.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Microsoft, Windows, and Windows NT are trademarks or registered trademarks of Microsoft Corporation in the U.S. and other countries.

All other names are used for identification purposes only and are trademarks or registered trademarks of their respective companies.

IKV++ GmbH
Informations- und Kommunikationssysteme
Bernburger Strasse 24-25
D-10963 Berlin
Germany
Email: ikv@ikv.de
URL: <http://www.grasshopper.de>

Contents

| | | |
|----------|---|-----------|
| 1 | Welcome to Grasshopper | 13 |
| 1.1 | Overview of Release 2.2 | 14 |
| 1.1.1 | New Features | 14 |
| 1.2 | Release Compatibility | 15 |
| 1.2.1 | Graphical Interface Rearrangement | 15 |
| 1.2.2 | Code Compatibility | 15 |
| 1.2.3 | Updating Grasshopper | 15 |
| 1.3 | Used Conventions..... | 16 |
| 1.4 | How to Get in Contact..... | 18 |
| 1.5 | Grasshopper Documentations..... | 19 |
| 2 | Grasshopper Installation | 21 |
| 2.1 | Before You Start the Installation..... | 21 |
| 2.1.1 | Hardware Requirements..... | 21 |
| 2.1.2 | Software Requirements | 22 |
| 2.2 | Installation of Grasshopper | 24 |
| 2.2.1 | Information Phase | 25 |
| 2.2.2 | Requesting Information | 28 |
| 2.2.3 | Installation Phase | 32 |
| 2.3 | Directory Structure | 35 |
| 2.4 | Testing the availability | 39 |
| 2.5 | From Here | 39 |
| 3 | First Time Invoking Grasshopper..... | 41 |
| 3.1 | Launching Assistant | 41 |
| 3.1.1 | Initialization of User Environment | 42 |
| 3.1.2 | Selecting an Application to Be Launched..... | 44 |
| 3.1.3 | Selecting Profile to Be Used | 45 |
| 3.1.4 | Launching an Application..... | 48 |
| 3.2 | Launching an Application via Command-line | 52 |

| | | |
|----------|--|-----------|
| 3.3 | From Here ... | 53 |
| 4 | Grasshopper Window System | 55 |
| 4.1 | Starting and Exiting Grasshopper | 55 |
| 4.1.1 | Starting a Region Registry..... | 55 |
| 4.1.2 | Starting an Agent System | 56 |
| 4.1.3 | Exiting an Agent System | 56 |
| 4.1.4 | Exiting a Region Registry..... | 57 |
| 4.2 | Exploring Grasshopper Windows | 57 |
| 4.3 | Agent System Features..... | 59 |
| 4.3.1 | The Menu Bar Interfaces | 59 |
| 4.3.1.1 | Agent Catalog..... | 67 |
| 4.3.1.2 | Console..... | 71 |
| 4.3.2 | The Toolbar | 73 |
| 4.3.3 | The Tree View | 75 |
| 4.3.4 | The Property View..... | 77 |
| 4.4 | Region Registry Features | 77 |
| 4.4.1 | The Menu Bar | 78 |
| 4.4.2 | The Toolbar | 80 |
| 4.4.3 | The Tree View | 81 |
| 4.4.4 | The Property View..... | 82 |
| 4.5 | From Here ... | 82 |
| 5 | Grasshopper Usage | 83 |
| 5.1 | Starting-up Grasshopper Environment..... | 83 |
| 5.2 | Agent Life Cycle | 83 |
| 5.2.1 | Creating an Agent..... | 84 |
| 5.2.2 | Removing an Agent | 85 |
| 5.2.3 | Suspending an Agent | 85 |
| 5.2.4 | Resuming an Agent | 85 |
| 5.2.5 | Copying an Agent..... | 85 |
| 5.2.6 | Moving an Agent | 85 |
| 5.2.7 | Invoking Agent's Action | 85 |

| | | |
|----------|---|------------|
| 5.3 | Organizing your Agents using Places | 85 |
| 5.4 | Usage Examples | 85 |
| 5.5 | From Here | 85 |
| 6 | Customizing Grasshopper | 87 |
| 6.1 |Customizing Your Profile | 87 |
| 6.1.1 | System Properties..... | 89 |
| 6.1.2 | Protocol Properties..... | 97 |
| 6.1.2.1 | Adding a New Protocol Entry..... | 97 |
| 6.1.2.2 | Removing a Protocol Entry..... | 98 |
| 6.1.3 | Server Properties..... | 98 |
| 6.1.3.1 | Adding a New Server Entry..... | 98 |
| 6.1.3.2 | List Properties of the Server Entry | 98 |
| 6.1.3.3 | Removing a Server Entry..... | 100 |
| 6.1.4 | Place Properties..... | 101 |
| 6.1.4.1 | Adding Places | 101 |
| 6.1.4.2 | Modifying Place Properties | 102 |
| 6.1.4.3 | Removing Places..... | 102 |
| 6.1.5 | Agent Properties..... | 103 |
| 6.1.5.1 | Adding an Agent Entry | 103 |
| 6.1.5.2 | Modifying an Agent Entry | 105 |
| 6.1.5.3 | Removing an Agent Entry | 105 |
| 6.1.6 | Profile Structure | 106 |
| 6.1.7 | Modifying the Profile..... | 107 |
| 6.1.7.1 | Adding New Places..... | 109 |
| 6.1.7.2 | Adding New Agents..... | 110 |
| 6.1.7.3 | Adding Agency's Attributes | 111 |
| 6.2 | Modifying Preferences | 112 |
| A | Grasshopper Program Switches..... | 115 |
| B | Grasshopper TUI Commands | 123 |
| B.1 | Common TUI-Commands..... | 123 |
| B.1.1 | Close | 123 |
| B.1.2 | Exit..... | 124 |
| B.1.3 | Gc..... | 124 |

| | | |
|----------|--|-------------------|
| B.1.4 | Gui | 124 |
| B.1.5 | Help or ? | 124 |
| B.1.6 | Info..... | 125 |
| B.1.6.1 | Output of Info..... | 125 |
| B.1.7 | List | 128 |
| B.1.7.1 | The Filter Construct..... | 129 |
| B.1.7.2 | Output of List | 133 |
| B.1.7.3 | Example..... | 135 |
| B.1.8 | Remove | 136 |
| B.1.9 | Status | 136 |
| B.1.10 | Thread..... | 137 |
| B.1.11 | History and Related | 138 |
| B.1.11.1 | Modifying the History Files..... | 139 |
| B.1.12 | Trace | 140 |
| B.2 | Agent-System Specific Commands..... | 141 |
| B.2.1 | Create..... | 141 |
| B.2.2 | Copy..... | 142 |
| B.2.3 | Invoke | 143 |
| B.2.4 | Move..... | 144 |
| B.2.5 | Resume | 144 |
| B.2.6 | Suspend..... | 144 |
| B.2.7 | Reregister..... | 145 |
| B.3 | Region Registry Specific Commands | 145 |
| C | Grasshopper StubGen Switches..... | 147 |
| D | Troubleshooting..... | 151 |
| E | Acronyms | 153 |
| F | Index | Annex -155 |

List of Figures

| | |
|--|----|
| Figure 1: Start-up Window | 25 |
| Figure 2: Introduction Window | 26 |
| Figure 3: Welcome Window..... | 26 |
| Figure 4: License Information Window | 27 |
| Figure 5: Latest Information about Grasshopper Product | 28 |
| Figure 6: Asking Location Information Window | 29 |
| Figure 7: Browse Location Window..... | 30 |
| Figure 8: Installation Type Selection Window | 31 |
| Figure 9: Component Selection Window | 32 |
| Figure 10: Summary Window | 33 |
| Figure 11: Acknowledgement Window..... | 34 |
| Figure 12: Wizard has detected an earlier version of Grasshopper | 35 |
| Figure 13: Directory Structure on Window-based System..... | 35 |
| Figure 14: Profile Path Window | 42 |
| Figure 15: Task Progress Window | 44 |
| Figure 16: Grasshopper Application Choice Window..... | 45 |
| Figure 17: Profile Selection Window | 46 |
| Figure 18: Profile Configuration Window..... | 47 |
| Figure 19: Region Registry Window..... | 49 |
| Figure 20: Region Textual Interface..... | 50 |
| Figure 21: Agent System Window | 51 |
| Figure 22: Agent System's Textual Interface Window | 51 |
| Figure 23: Exit via Graphical User Interface..... | 57 |
| Figure 24: Exit Dialog Box..... | 57 |
| Figure 25: The Graphical Elements of Grasshopper Window..... | 58 |
| Figure 26: The Agent System Window | 60 |
| Figure 27: The Menu Bar Part 1 | 60 |
| Figure 28: The Preferences Dialog Window | 61 |

| | |
|--|-----|
| Figure 29: The Menu Bar Part II | 63 |
| Figure 30: The Place creation dialog box..... | 63 |
| Figure 31: The Agent Creation Dialog Box..... | 64 |
| Figure 32: The Location Specification Dialog Box | 64 |
| Figure 33: Agent Catalog Window | 65 |
| Figure 34: The Thread Monitoring Window | 66 |
| Figure 35: Agent Catalog Window | 67 |
| Figure 36: Preference Window of the Agent Catalog | 68 |
| Figure 37: Toolbar of the Agent Catalog | 69 |
| Figure 38: Agent Entry Window | 70 |
| Figure 39: Agent Catalog View part | 71 |
| Figure 40: Agent Console Window | 72 |
| Figure 41: Toolbar of the Console | 72 |
| Figure 42: Toolbar of the Agent System Window | 73 |
| Figure 43: Agent Creation Dialog Window | 74 |
| Figure 44: Graphical Elements of the Tree View..... | 76 |
| Figure 45: Context Menu within the Tree View | 76 |
| Figure 46: Property View of the Agent System Window | 77 |
| Figure 47: Region Registry Main Window | 78 |
| Figure 48: All Entries of the Menu Bar..... | 79 |
| Figure 49: Toolbar of Region Registry | 81 |
| Figure 50: Region Registry Tree View | 82 |
| Figure 51: Profile-Configurator for Agent System | 88 |
| Figure 52: Adding new Protocol to Grasshopper | 97 |
| Figure 53: Adding Servers | 99 |
| Figure 54: Server Entry Properties | 101 |
| Figure 55: Modifying Place Properties | 102 |
| Figure 56: Adding an Agent Entry | 105 |
| Figure 57: Removing an Agent Entry from Your Profile | 106 |
| Figure 58: Structure of a Profile..... | 108 |

| | |
|--|-----|
| Figure 59: The default Profile..... | 108 |
| Figure 60: myProfile..... | 109 |
| Figure 61: The Agency using myProfile | 110 |
| Figure 62: myProfile including Agent entries | 111 |
| Figure 63: Extending Agency's Entry | 112 |
| Figure 64: The Preference Window..... | 113 |
| Figure 1: Detail Information Related to a Place | 126 |
| Figure 2: Detail Information Related to an Agent..... | 127 |
| Figure 3: Detail Information Related to an Agent-System..... | 127 |
| Figure 4: List Command..... | 128 |
| Figure 5: List Output Description..... | 134 |
| Figure 6: Outputs of the „Thread“-Command | 138 |
| Figure 7: History Command | 139 |
| Figure 8: Copy | 143 |
| Figure 9: Provided Region-Tui Commands..... | 146 |

List of Tables

| | |
|--|-----|
| Table 1: Notational Conventions | 16 |
| Table 2: Visual Elements..... | 17 |
| Table 3: Grasshopper Hardware Requirements..... | 21 |
| Table 4: Programs in the Bin-Directory | 36 |
| Table 5: Grasshopper System Libraries..... | 38 |
| Table 6: Common System Properties | 89 |
| Table 7: General Security Settings | 92 |
| Table 8: Common SSL Security Properties..... | 93 |
| Table 9: Agent Specific Properties | 94 |
| Table 10:Agent Related Security Properties | 96 |
| Table 11:Region Specific Properties | 96 |
| Table 1: Command Flags..... | 116 |
| Table 2: Option Flags | 117 |
| Table 3: Parameters Options..... | 119 |
| Table 4: Filter Keys | 130 |
| Table 5: Comperators | 132 |
| Table 6: Arguments of the Trace Command | 140 |
| Table 7: Command-line Options to Stubgen | 148 |

1 Welcome to Grasshopper

Grasshopper is an agent development platform that enables you to exclusively develop and deploy a wealth of distributed, agent-based applications written in the Java programming language. This platform provides to you new opportunities for the enhancement of electronic commerce applications, dynamic information retrieval, advanced telecommunication services and mobile computing.

Because Grasshopper makes you possible to build software agents to move between different systems, allowing location independent computation, your applications benefits of the advantage of local high-speed communication and local high-speed access.

In detail, Grasshopper enables you to:

- Create autonomous acting agents, which are able to migrate
- Transparently locate agents and send messages to them
- Manage and control agents during their execution by providing sophisticated tools
- Provide a wealth set of example agents that can be used as a starting point for exploitation of agent technology.

By providing add-ons for Grasshopper, creating agent-based applications and services which are interoperable with other agents systems that are compliant to the OMG Mobile Agent System Inter-operability Facility (MASIF: www.omg.org/docs/orbos/97-10-05) or Foundation of Intelligent Physical Agents (FIPA: www.fipa.org) are enabled, ensuring your investment into the implementation for the future.

The remaining sections of this document provide a brief overview of MA concepts and focus on the installation and usage of the Grasshopper development system via its command line and graphical user interfaces. More information on concepts can be found in **Basics** and **Concept**. Issues related with agent programming matters are provided in the **Programmer's Guide**.

Parts of Grasshopper Product

If you have down loaded Grasshopper from our Web-site, you will probably have the self-extracting program, which will install Grasshopper software on your system and the respective documentation.

1.1 Overview of Release 2.2

In this section you will find a brief description of each new or improved feature in Grasshopper release 2.2.

1.1.1 New Features

JDK 1.2 and 1.3 Support

Grasshopper supports the latest in Java technology. You can create, compile, and deploy agent code that complies with the JDK (Java Development Kit) 1.2. or now also known as Java 2.

You can also choose to use the latest version of the JDK 1.3. Note that if JDK 1.3 is used, there is no need to generate proxies for agents with the provided *Stubgen* tool.

There is in addition a Grasshopper runtime version for Windows CE. You will have to download a separate installation program from the IKV website.

To use either one of the JDK versions, you have to download them from Sun Microsystem's Web site (<http://java.sun.com>).

The new benefits in short of Grasshopper 2.2 are:

- The textual user interface of the Agent System and the Region Registry has been advanced with new powerful functions, allowing you to operate Grasshopper without graphical supports.
- The application programming interface of Grasshopper has been reworked, making it more intuitive than ever. Also new interfaces have been added, extending functions of agents.
- The graphical user interface of the Agent System has been improved, enabling a more effective life cycle control of agents.
- The core engine of Grasshopper has been improved in respect to the execution performance, reducing the reaction and communication time drastically.
- An improvement of the agent handling system, making it more scalable.
- The support LDAP as a Domain-Server has been integrated into Grasshopper.

1.2 Release Compatibility

This section gives an outline of issues related to compatibility of different release versions.

1.2.1 Graphical Interface Rearrangement

If the most recent release of Grasshopper you are accustomed to using is before release 1.2, you will become aware of the entire new graphical user interface (GUI) of release 2.2. You will soon experience that the new GUI is more user-friendliness compared to the previous one and is more adjusted to the characteristics of agent-based applications, thus, allowing you a more intuitive usage and control.

1.2.2 Code Compatibility

The new release 2.2 is a new designed and implemented development system that has internally nothing in common with the releases prior to 1.2. These significant changes were necessary in order to improve the performance and stability of Grasshopper. In this connection, some redundant interfaces were omitted. Incompatibility of code will exist, where functionality and interfaces have changed between different releases, making them not runnable in release 2.2. Also other incompatibilities are due to the shift from JDK 1.1 to JDK 1.2.

So basically you agent applications developed with version 2.2 beta and 2.1 should be source and binary compatible. Applications developed with 2.0 and older have to be changed at source code level.

However and in general, a recompilation of the code implemented with any previous versions is recommended.

1.2.3 Updating Grasshopper

IKV provides from time to time patches and updates to Grasshopper. You can download them from Grasshopper's Web site (www.grasshopper.de) via the Internet by using a common browser, such as, Netscape or Internet Explorer.

If you don't have access to the Internet or your access is restricted by firewalls, you can mail or call our Support Department by using the below information.

IKV++ GmbH

Grasshopper Support
Bernburger Strasse 24-25
10963 Berlin, Germany
Email: grasshopper2@ikv.de
Phone: +49 30 34807716

1.3 Used Conventions

This documentation contains a variety of special conventions to help you find information you need, fast. Formatting conventions are used in order to emphasize important keywords or text passages. A right dose of visual elements are used to make substantial and helpful information obvious. The following sections describe all of these used elements in the User's Guide.

Formatting Convention

The User's Guide uses typeface conventions to help you understand what you are reading:

| Convention | Description |
|---|--|
| Proportional Font | Used for standard text |
| <i>Proportional Italic Font</i> | Used either to emphasise words or to indicate the first appearance of new terms. |
| Fixed Font | Used to identify source code, email addresses and http addresses. |
| <i>Fixed Italic Font</i> <same font in brackets> | Indicates commands that the user has to type. Indicates parts of the command that the user has to substitute by concrete values, e.g., a file name. |
| Fixed Bold Font | indicates on-screen messages or comments within a source code listing. |

Table 1: Notational Conventions

Visual Elements

A set of diverse visual elements give you useful information and draw your attention to topics of specific interest. These elements are placed at the page

margins:







| Elements | Descriptions |
|---|--|
|  | This icon indicates information that is specific for Unix operating systems. |
|  | This icon indicates information that is specific for Windows operating systems. |
|  | This icon indicates paragraphs that provide some background information about a specific topic. This information is not required for the understanding of the respective section and may be skipped by the reader. However, it may be interesting for readers who want to know more about the concepts of the Grasshopper. This background information is additionally highlighted by means of a shaded frame. |
|  | This icon indicates useful tips and tricks that facilitate the usage of the product. |
|  | This icon indicates paragraphs that are of particular importance and that should be read in any case, even if you want to go through the document as fast as possible. |
|  | This icon is used to indicate examples. |

Table 2: Visual Elements

Chapter Roadmaps

Each chapter begins with a brief introduction and a list of topics you will find covered in that chapter. So you will always know in advance what you are going to read about before you start.

Graphical User Interface Interactions

Many sections within the User's Guide describe specific operations that a user can perform via the graphical user interface (GUI), such as agent creation or termination. In order to enable you to learn the required interaction steps quickly and without reading thru the whole section, each section starts with a single line, containing the menu and menu item(s) which are required to per-

form the desired operation. These lines are emphasized by a surrounding frame.

Example: To view or modify the preferences of an agency (cf. Chapter 5), the user has to select the item Preferences of the agency's File menu. After this action, the Preferences window appears. These interactions are summarized by means of the following line at the top of the corresponding section:

File -> Preference

There are also some conventions in using the mouse:

- Left button (single click): A single click on the left mouse button activates menu items or selects entries in tree browsers or tables. Note that this is the "default" mouse action in the context of this manual. Thus, the instruction "Select the agent entry" requests the user to move the mouse pointer onto the mentioned entry and to perform a single click, using the left mouse button.
- Left button (double click): A double click on the left mouse button invokes specific actions on GUI elements. For example, a double click on agent entries within the Agency Console invokes the Action() method of an agent. Note that the result of a double click is explicitly explained for every interaction.
- Right button (single click): A single click on the right mouse button activates pop-up menus. For example, moving the mouse pointer onto an agent entry within the Agency Console and pressing the right mouse button activates a pop-up menu that enables the user to suspend, resume, copy, clone or remove the agent.

1.4 How to Get in Contact

If you ever have to make suggestions, critics, or even compliments about Grasshopper, please feel free to send an email to:

`grasshopper2@ikv.de`.

Our team will look at any incoming information from every angle.

You can find additional information from our web sites:

`www.grasshopper.de`.

In order to retrieve the comments of other Grasshopper users and participate

in discussions, please visit our Web site:

<http://www.grasshopper.de/community>

and subscribe to the discussion groups you are interested in.

1.5 Grasshopper Documentations

Grasshopper comes along with a set of documentations guiding you to different topics of the usages as well as providing concepts of agent technology in general. You will find the following documentation parts:

- **Basics and Concepts.** This part comprises an introduction to mobile agent technology and to the Grasshopper development platform.
- **User's Guide.** This part describes the platform installation and its usage via graphical and command line interfaces.
- **Programmer's Guide.** This part explains how to realise mobile and stationary agents on top of the Grasshopper Development System.

Structure of the User's Guide

The User's Guide gives you direct answers about how to use Grasshopper. It is organized to follow the following cycles:

CHAPTER 1, Welcome to Grasshopper, this part of the document, gives an overview of this manual, its background and an outline.

CHAPTER 2, Installation, describes in detail how to install Grasshopper.

CHAPTER 3, Getting Started, describes an outline how to start and use an agency and a region registry by introducing a brief example.

CHAPTER 4, Exploring Window System, explains all graphical components of Grasshopper.

CHAPTER 5, Using an Agency, explains how to start an agency via the graphical front-end and via command line. It also provides in detail the complete management functionality that is provided via graphical user interfaces. Besides, you can find a usage description of the example agents which are part of the Grasshopper release.

CHAPTER 6, Using Region Registry, describes how to start a region registry via the graphical front-end and via command line.

CHAPTER 7, Customizing Grasshopper, describes all possibility you have to customize Grasshopper for your particular needs.

ANNEX A, Grasshopper Program Switches, identifies and explains all com-

mand-line parameter you can apply to the Grasshopper start-up program.

ANNEX B, Grasshopper TUI Commands, an agency and a region can be started with a textual interface. The available commands and their usage are described within this annex.

ANNEX C, Profile Structure, this appendix illustrates the structure of the profiles used by Grasshopper, and it explains how you can modify them directly with an editor.

ANNEX D, Troubleshooting, will list some known problems and explain how you can omit them.

ANNEX B, Acronyms, a list of used arconyms are outlined in this part.

ANNEX C, Index, a list of keywords of the User's Guide can be found here.

Screenshots

All screen shots in this document are taken from the Windows NT version of Grasshopper. Compared to the Unix version, there may be minor differences regarding the window decoration, depending on the used window manager.

2 Grasshopper Installation

This section contains information for installing Grasshopper 2. It describes the steps you must take and information you must prepare prior to installing Grasshopper.

This section will include:

- Description of the installation requirements of Grasshopper
- Instructions and guidelines to install Grasshopper on Windows9x, Windows NT 4.0 SR3, Windows 2000, SUN Solaris 2.5/2.6 and Linux.
- Outlines the steps to remove Grasshopper
- Provides troubleshooting information for your installation

2.1 Before You Start the Installation

Prior to any installation attempt, you shall make sure that your system fulfils the hardware requirements of Grasshopper. Additional, in order to run the Grasshopper software installation program successfully, you must have the privileges for installing software on your local machine. At last, it is also recommended to remove previous versions of Grasshopper before installing this release.

2.1.1 Hardware Requirements

To use the Grasshopper's software installation routine, you shall ensure that the targeting system fulfils the hardware requirements mentioned in Table 3.

| Items | Requirements |
|------------------|---|
| Operating System | Grasshopper is completely developed in Java. Theoretically, it runs on operating system where a Java Virtual Machine (JVM) is available. Grasshopper has been tested on SUN Solaris 2.5 and 2.6, Windows NT 4.0 SR5, and Windows 9x |

Table 3: Grasshopper Hardware Requirements

| Items | Requirements |
|------------|--|
| Memory | Windows 9x:32 min.,64 recommended Windows NT:48 min.,80 recommended Windows 2k:48 min.,128 recommended Solaris: 64 min.,96 recommended |
| Disk Space | Max: 5 MB, Min. 2 MB |
| Processor | Since Grasshopper is completely developed in Java, it runs theoretically on any system where a Java Virtual Machine (JVM) is available. Grasshopper has been tested on Sparc Processor and PC-compatible computer system (Intel, AMD). |
| Others | It is recommended to run Grasshopper on a system with a graphical display (256 or colors) and any pointing device with at least two buttons or comparable (Apple MAC) in order to use the graphical user interface. |

Table 3: Grasshopper Hardware Requirements

2.1.2 Software Requirements

Grasshopper itself is a self-contained software package. Even though, it requires software packages from third party vendors. Some of them are mandatory while others are just optional, depending on the desired functionality you need.



However, the CLASSPATH environment parameter must contain entries to these installed packages in order to enable Grasshopper to use them. For more information about the installation procedure of these software packages, please refer to their respective manuals.

Java Runtime Environment (JRE)

Grasshopper is fully implemented in Java. It therefore requires the installation of a Java runtime environment. Grasshopper has been tested with JRE 1.2 from Sun Microsystem Inc. (www.javasoft.com). If you want to use Grasshopper in the most convenient way, you shall install version 1.3. For more details please refer to the Programmer's Guide.

Others JREs may also be used, but you have to make sure that they are fully compliant to SUN's Java Specifications.

Note, in order to install Grasshopper you need a fully installation of a JRE. Why? This is just because the Grasshopper Installation program itself is entirely written in Java.



Corba Runtime Environment

In contrast to the previous version, you are not confronted with the installation of a particular CORBA 2.0 runtime environment. These steps has become obsolete since the JRE 1.2 or higher is shipped with a free CORBA Orb, which is used as the default Orb. Nevertheless, if you require to use an Orb from another vendor, such as from IONA Technologies PLC, the respectively software package has to be installed. In order to enable Grasshopper to use this ORB, you have to make sure that Java always find this software package first instead of the one from SUN's JRE. One way to do this is to modify the CLASSPATH environment parameter in such a way that the entry pointing to the software package is prior to the one for SUN's JRE.

Please note, Grasshopper has only be tested successfully with IONA's Orbix-Web 3.1 and VisiBroker 4.0. IKV++ makes no warranty that other Orbs can be used successfully. Nevertheless, in the Programmer's Guide some steps are described how you can modify Grasshopper in order to use another Orb.



Also, for example, some Orb vendors have foreseen the usage of JRE 1.2 and have taken this into consideration for their products. In this case, please refer to their provided documentation.

Security Runtime Environment

Grasshopper also provides an excellent security support for applications, such as, required for banking or e-commerce sector. Along several authentication features, different granularity levels of access control are provided. In order to activate some of the security mechanism (i.e. SSL or additional security algorithm), you have to install the third party's security packages. Grasshopper currently supports 2 SSL packages

IAIK, which are provided by the IAIK-Java Group, that is part of the institute for applied information processing and communications (IAIK) of the technical university of Graz. More information on the IAIK product line and how to download and buy the software can be found on their Web-site: <http://jcewwww.iaik.tu-graz.ac.at>.

JSSE, which is Sun's Java Secure Socket Extension. More information can be found on their Web-site <http://java.sun.com/products/jsse/>.

Tests has been conducted successfully with the IAIK 3.0 Release and JSSE 1.0.2.

Once you have purchased/downloaded the software and have installed the security packages, i.e., the CLASSPATH environment parameter is enhanced with entries pointing to the respective Java classes or archives, Grasshopper will detect the availability of these classes and will enable the security features of the platform.

For information about additional installation steps, please refer to the respective manuals or contact directly the support line of the university of Graz or Sun. Information of contacts points can be found on their Web-site.

Viewing Editors

There is also a set of documentation where you can learn the different aspects of using Grasshopper software and/or programming your agents. These declarations are either included in the software package or can be explicitly downloaded from our Web-site.

For viewing the application programming interfaces you need an HTML-enabled viewer, such as the Netscape Browser (www.netscape.com) or Microsoft Internet Browser (www.microsoft.com). These can be downloaded from the respective sites without any extra cost.

2.2 Installation of Grasshopper

Grasshopper comes as a self extracting program, which will guide you thru the installation process. This program, the so-called setup wizard, is written in Java and thus, a JRE must be installed previously.



To start the installation on Windows9x/NT/W2K, you have to use Grasshopper.exe. This program can be downloaded from our Web-site www.grasshopper.de.



To install Grasshopper on Unix system, the Grasshopper.sh shell script has to be executed. The script will launch the wizard and will assist you in the installation process. The script can be downloaded from our Web-site www.grasshopper.de.

The installation process can be generally divided into three phases:

- The first phase, the information stage, provides some additional information, such as license agreement and latest information about the product, which are not included in the handbooks (see Section 2.2.1).
- At the second phase, the requesting information stage, the wizard is asking you about some information related to the installation process (see Section 2.2.2).

- At the final, the wizard is doing the actual installation for you on your system (see Section 2.2.3).

But before this and once you have executed the wizard, either with Grasshopper.exe on Window-based system or Grasshopper.sh on Solaris system, you will be asked whether you want to start the installation process.

On the Windows-based system, the following graphical component shall be visible to you on your system's screen (see Figure 1).



Figure 1: Start-up Window

On a Unix system with no graphical interface a text based installation is possible. Asking you whether you like to install Grasshopper or not is being done within the shell from where you have started the wizard.



Once you have given a positive conformation, the wizard will analyse your system whether it fulfils the installation prerequisite as described in the previous section.

Please note that from this point on, there are no graphical differences between a Windows-based and a Solaris-based system, since the wizard is written in Java and the graphical components are based on Swing. A slight distortion may result in the different font types and font handling routines of both systems.



For the following presentation, all the graphical components are taken from a Windows NT system.

2.2.1 Information Phase

Once you have applied the „Yes“-button, an introduction window is shown, while the wizard is checking your system.

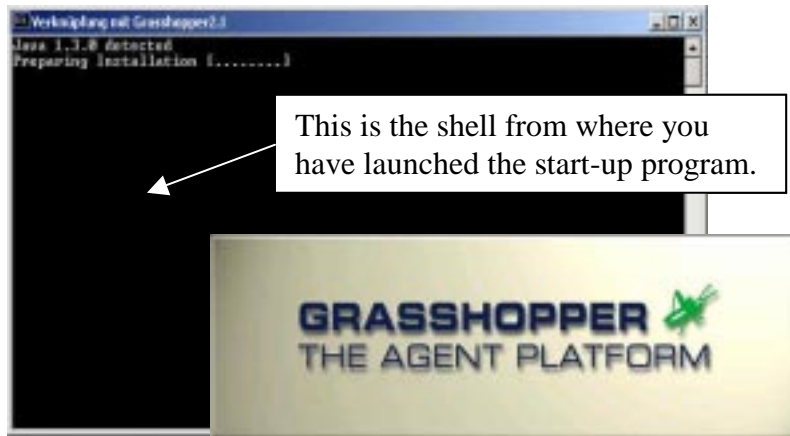


Figure 2: Introduction Window

Thereafter, i.e., your system has been successfully tested, the wizard will welcome you by showing the splash screen as pictured in Figure 3. No particular purposes is behind the window except for your information.

You shall read the provided information to legal aspects related to reproduction and distribution of Grasshopper software to a third party carefully. Just keep this in mind and you can continue with the installation without any consideration.



Figure 3: Welcome Window

Besides the information on unauthorized distribution and reproduction, some licensing information are presented in the subsequence window (see Figure 4)



Figure 4: License Information Window

The newest information related to Grasshopper can be found on the next window. It contains last-minute product information, updates to the Grasshopper documentation and/or troubleshooting tips. Any information mention in this window will supersede any written information provided along with Grass-

hopper. The Figure 5 shows this window.

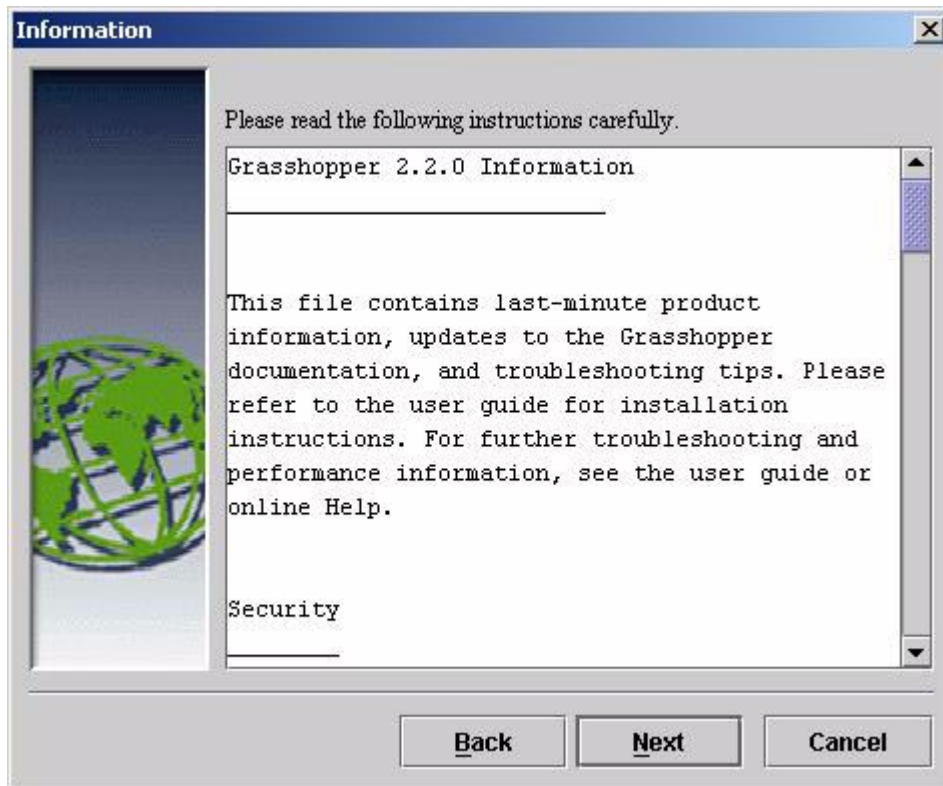


Figure 5: Latest Information about Grasshopper Product

2.2.2 Requesting Information

The wizard needs some information from you in order to install Grasshopper onto your system. First, you have to tell where the wizard shall put the Grasshopper software packages. This is requested via the following window (see Figure 6).



Figure 6: Asking Location Information Window

The wizard will suggest a default location, which is determined by using the following hard-coded schema:

On a Windows-based system, the wizard will always prompt: C:\Program Files\Grasshopper.

The default location for storing Grasshopper's software packages on a Unix system is /usr/local/Grasshopper.

You should make sure that you have the privilege to install software within the suggested locations.

In case you wish to provide another location, use the „Browse“-button. Once pressed, the following window will show up on the screen (see Figure 7).

Due to limited capabilities of the Swing packages, you are not directly able to type in the folder name within the text-field. You have to explicitly choose the folder with the mouse pointer and press your right mouse-button.





For example, in order to create a new folder, let's say „gh“, you have to do the following steps:

- First, press on the „Create New Folder“-Icon. This will create a new folder at the current directory location. This folder is automatically named as „New Folder“.
- Next, you have to select this newly created folder by moving your mouse pointer to it and by pressing the selection button of the mouse. Once selected, the folder shall be emphasized. Also, the name of the selected folder shall appear within the „Select Folder name“-Field.
- Thereafter, double-click on the selected folder. This will activate the modification mode, which allows you to over-write the existing folder name. Now, please type in „gh“.
- The last step for selecting the new folder name is just by selecting the folder „gh“.

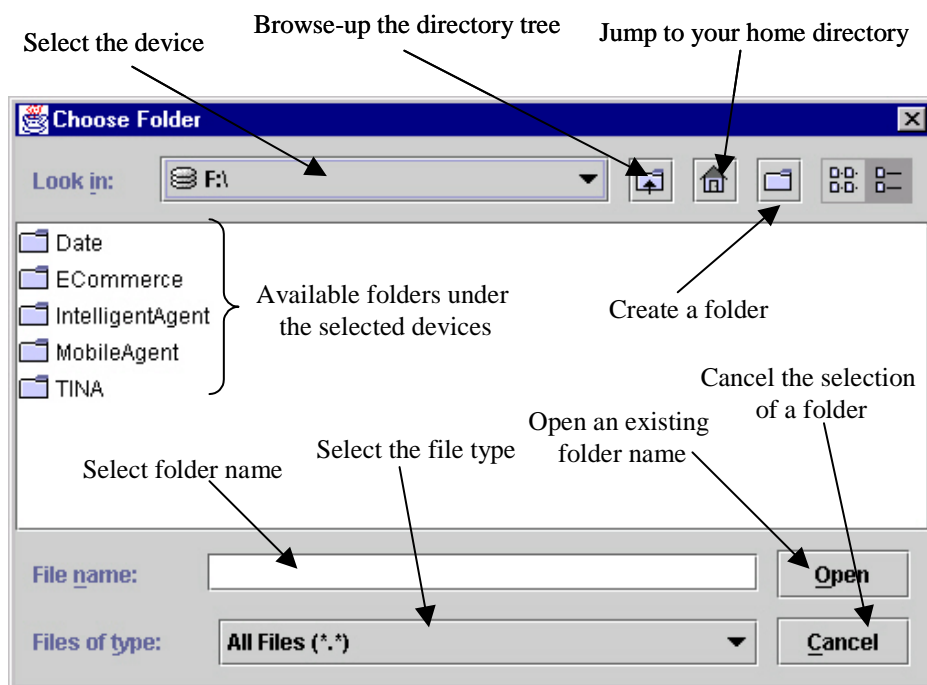


Figure 7: Browse Location Window

Once you have selected the folder to be used by the wizard, the dialog will return to the previous screen, where you can finishing up the folder selection activity.

After pressing the „Next“-button, you have to specify the type of the software

installation. Three types are provided as shown in Figure 8.



Figure 8: Installation Type Selection Window

- The first one, the compact type, will install the bare minimum file to run Grasshopper. This type will reduce the required space on your device. The examples and documentation will not be installed.
- The second one, the custom type, allows you to select the software components to be installed. This is recommended for an advanced user, only.
- The last one, the typical type, will install the same as the first one, including documentation and examples.

Only if you decide to use the custom type of installation mode, a component selection window (see Figure 9) will be shown to you, enabling you to select only those components to be stored on your system. Otherwise, you will proceed directly to the installation stage which will describe at the Section 2.2.3.

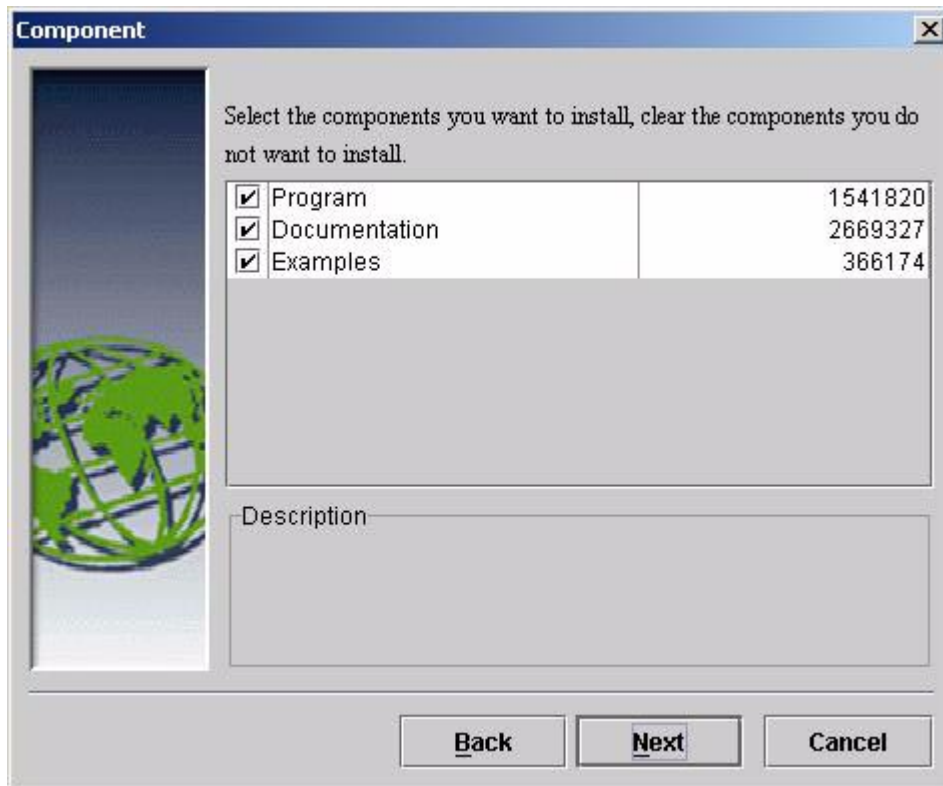


Figure 9: Component Selection Window

2.2.3 Installation Phase

After you have selected the installation type, the wizard will provide a summary window (see Figure 10) containing all the installation related information you have passed in. After checking the correctness, you give the wizard the sign to start the actual installation of Grasshopper on your system by pressing the „Next“-button.

If you have detected some misspelled or incorrect information, you can navigate back to the respective window for modifying your inputs by using the „Back“-buttons.



Figure 10: Summary Window

The wizard will provide you a report whether the installation process were successfully carried out or not. This is done by showing the respective ac-

knowledge window as shown in Figure 11.

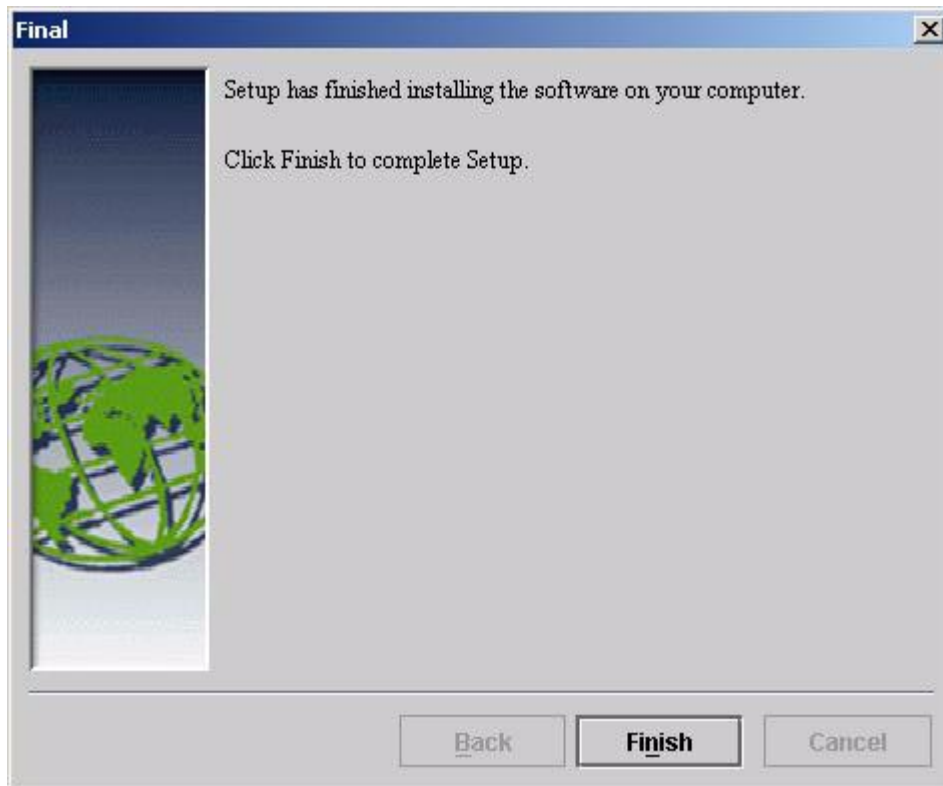


Figure 11: Acknowledgement Window

However, in case the wizard detects some error during the installation process, some respective windows will also be shown, providing instruction for solving the identified problems.

For example, if you have installed a previous version of Grasshopper and have not un-installed it, the wizard will detect this during the installation process and will display the following windows to you (see Figure 12).

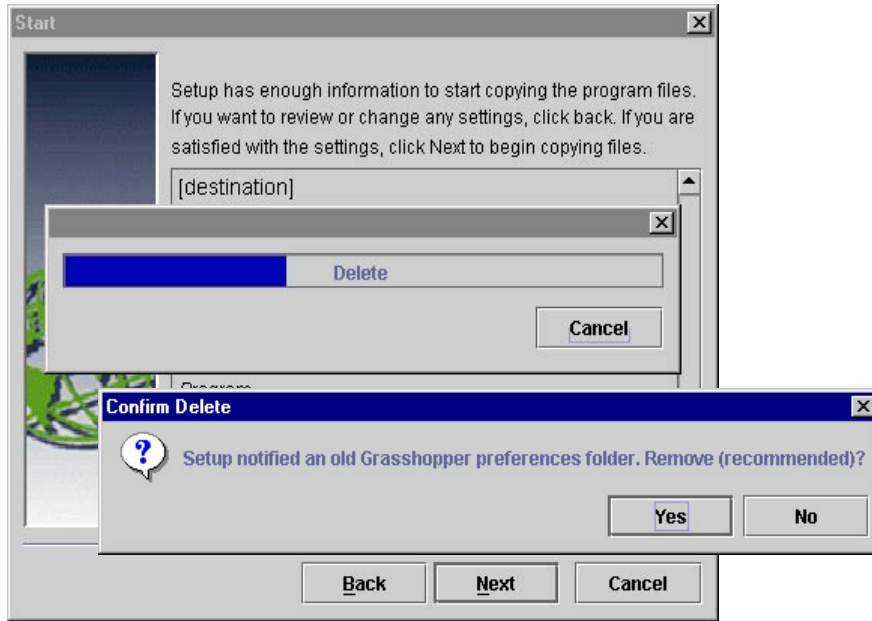


Figure 12: Wizard has detected an earlier version of Grasshopper

2.3 Directory Structure

After installing Grasshopper, the following directory structure shall have been created by the wizard:

On a Windows-based system, the directory structure shall look like the one as depicted in Figure 13.



Figure 13: Directory Structure on Window-based System

On a Unix system, the directory structure shall look like the same depicted in Figure 13.



Root Directory

Remember the name of the root directory? Right, it is exactly the name you have provided during the installation process as described in the previous sections (c.f. Section 2.2.2).

Besides the portrayed folders, which will be further introduced about their contents later in this section, you will find the licence agreement of Grasshopper in the root directory. You shall read this carefully and always keep the content in your mind, whenever you use Grasshopper software or distribute it to a third party.

Bin Directory

The Bin-Directory contains some executables of Grasshopper. These executables are either a shell script, e.g for Solaris system, or a batch file, for Windows-based system. The purpose of these executables is described in Table 4.

You shall include the search path of your system with the Bin-directory. This makes it more convenient to starting up Grasshopper.



| Programs | Purpose |
|------------------|--|
| Catalog.bat | The Catalog.* script launches the agent catalog application in standalone mode. Please see the online documentation for further details. |
| Catalog.sh | |
| Explorer.bat | The Explorer.* script launches the Grasshopper explorer in standalone mode. You can use this script to attach the Explorer to an already running agent system. |
| Explorer.sh | |
| Grasshopper.bat | The Grasshopper.* executable represents the start-up program. The extension of the file tells you for which system it can be used. So, e.g., the *.sh-extension is for Solaris while the *.bat-extension for Windows-based system. |
| Grasshopper.sh | |
| PolicyEditor.bat | The PolicyEditor.* script launches the enhanced place-based policy editor coming with Grasshopper. Please see the online documentation for further details. |
| PolicyEditor.sh | |

Table 4: Programs in the Bin-Directory

| Programs | Purpose |
|-------------------|---|
| SecurityCheck.bat | The SecurityCheck.* script is a simple tool to list all security providers and algorithms, currently supported by your Java environment. |
| SecurityCheck.sh | |
| StubGen.sh | Analog to Grasshopper.*, there are also two programs for generating stubs suitable for the grasshopper's agent system for Solaris and Windows-based systems. More Information on the usage of the Stub can be found either in Annex C or in the Programmer's Guide. |
| StubGen.bat | |
| SystemCheck.bat | The SystemCheck.* script is a simple tool to list all major information about your Java environment like OS or JVM type, version and more. |
| SystemCheck.sh | |
| Tools.bat | The Tools.* script starts a wizard which provides you with a set of convenient tools. They are unsupported und currently not documented but could be used as they are. |
| Tools.sh | |

Table 4: Programs in the Bin-Directory

Doc Directory

Beneath the Doc-directory, you will find the application programming interface, or shorthand to API. In order to view this specification, an HTML-enabled viewer must be installed. All API-specifications can be found in the API-directory.

If no such a viewer is installed, please consult you local system administrator for installing such or acknowledging it from the respective Web-site (see Section 2.1 for site information).

Examples Directory

Besides the manifold provided documentation, there is also a comprehensive set of examples that make your tasks of learning agent programming much easier.

Most of the examples are used in the Programmer's Guide, thus you will find more information on them at the respective passage. You can use the provided sources to learn the programming pattern of Grasshopper agents.



Make sure that the path to the examples is included in your CLASSPATH environment, in order to test some of them.

Lib Directory

The Lib-directory contains the class files, which are put into a JAR-file, of the Grasshopper system. The following Table 5 describes the purpose of each JAR-files.

| JAR-File | Purpose |
|----------|--|
| gh.jar | This file contains the core class files of the Grasshopper system. It is necessary that you append this jar-file to your CLASSPATH. In contrast, the other mentioned jar-files are optional. |
| jndi.jar | If you want to use the Java Naming Directory Interface (JNDI), you have to include this, ldap.jar and providerutil.jar to your CLASSPATH. This JAR-file represents the client part of accessing the directory service. Also, note that JNDI is not required if you are using JAVA Version 1.3. |
| ldap.jar | This file implies the classes for using a LDAP-server at your premises. Please contact your local administrator for more detail on this issue. If you are using JAVA Version 1.3. this file does not have to be part of the CLASSPATH. Also, the jndi.jar and providerutil.jar have also be included in the CLASSPATH. |

Table 5: Grasshopper System Libraries.



In order to use, these JAR-files must be included in your CLASSPATH environment parameter. Remember, since these are JAR-files, the name of each file must be explicitly appended to the CLASSPATH.

2.4 Testing the availability

Once you have installed Grasshopper and have set the required configuration, such as extending the CLASSPATH with the Jar-files and including the Bin-directory to your search path, you can make a short test to see whether you are able to start-up Grasshopper.

First, start a command shell and type in at the command-line: Grasshopper. If you have followed the installation step by step you will see a welcome window display on your screen.



In case, some errors occurred please check whether you have strictly followed the installation instructions, and/or that your system fulfilled the hardware as well as the software requirements as stated in Section 2.1.

2.5 From Here ...

The sections of this chapter have explained and described the steps to install Grasshopper on your system.

From here, you can directly go to the next chapter, which will guide you through the different applications of Grasshopper and explain the usage of them.

If you are an advanced user, you can go directly to the chapters dealing either with the usage of an Agency, Agent or Region.

3 First Time Invoking Grasshopper

Right after the successful installation you are able to launch Grasshopper immediately. This section will describe all required procedures for starting up Grasshopper. Please follow the presentation thoroughly. This will grant a successful settings of your Grasshopper version.

There are several ways to start-up Grasshopper applications, either an Assistant, Agency or a Region Registry. These applications can be started by means of command-line instructions. You can also use the graphical user interface of the Assistant for launching an Agency or Region.

The first section will introduce the usage of the Assistant, while the second one will describe the activation of Grasshopper's applications via command-line options.

3.1 Launching Assistant

Grasshopper provides an Assistant for guiding you through the starting procedures of Grasshopper. This Assistant can be launched by the program called „Grasshopper.*“, which is included in the Grasshopper product. You can find this program in the sub-folder bin of the directory where Grasshopper was installed, the home directory of Grasshopper. Please refer to Chapter 2.2.2, where you can find information about the installation directory.

Many options can be passed to the „Grasshopper“ program. A detail description of them can be found in Annex A.

To activate this program on a Microsoft Window system (NT or Win9x), there are many alternatives. One possible way is for example:

by starting an MSDOS-shell and type in on a command-line `%GHHOME%\bin\Grasshopper*`. `%GHHOME%` is virtual and is just a place holder, pointing at the installation directory of Grasshopper. You have to set this holder to the actual installation path. This holder becomes obsolete if the „PATH“ environment includes the path of the Grasshopper bin directory.

On a Solaris System just run the script „Grasshopper“. The mentioned Class-path setting and Path also holds in a Solaris system.



3.1.1 Initialization of User Environment

Grasshopper can be run in a multi-user environment. Each user can individual customize the Grasshopper platform. This custom information is stored in a configuration files, named simply as „Profile“.

With such a „Profile“, you can provide your settings and preferences related to an Agency, a Region Registry or an Assistant. This information will then be taken by the Assistant and be used for the start-up process of Grasshopper's applications.

Profile Window

The „Profile“ window, as shown in Figure 14, will appear after you have pressed the „Next“-button at the „Welcome“ window. This window displays the default path used by the Assistant to locate the profiles related to the user that has launched the Assistant. For example, Figure 14 shows a path in a Microsoft Window NT system. On a solaris system, an equivalent path will be shown.

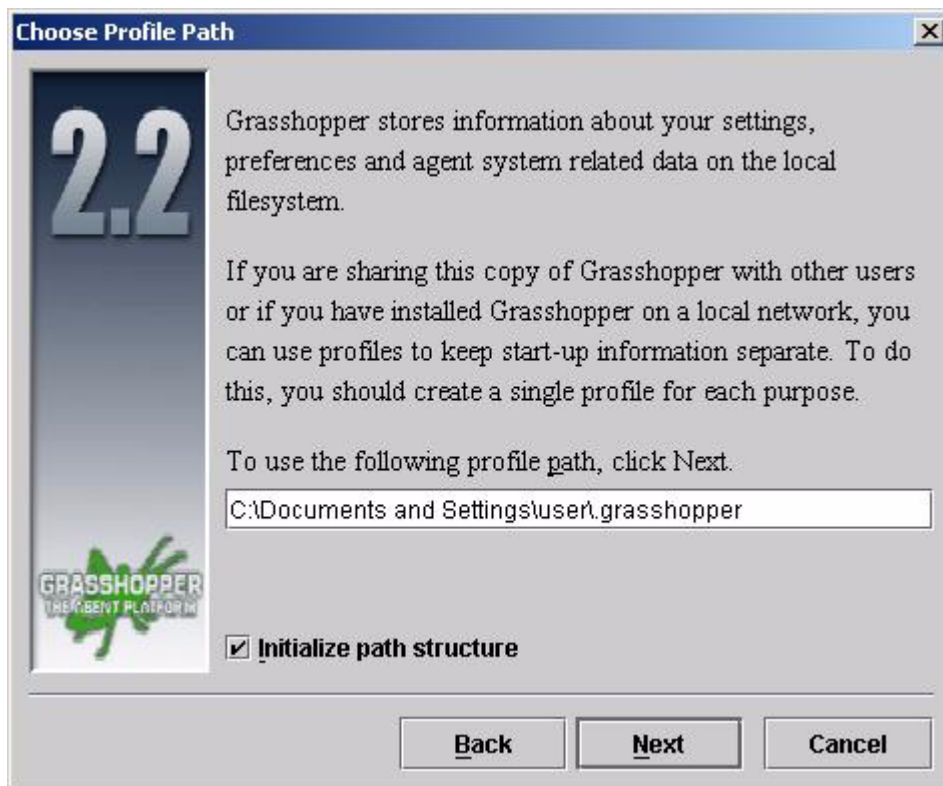


Figure 14: Profile Path Window

Setting up the Profile Path

In default, the Assistant assumes that each user has a user home directory. This home directory is then used by the Assistant to store the user's profile. Depending on the system you are using, the path specification may look differently.

On Microsoft Windows NT, the path specification usually starts with a disk drive letter, followed by the installation path of Windows NT and the subdirectory Profiles. However, each element of the path may vary from system to system.



On Solaris system, the path does not contain a disk drive letter. Usually, the user's home is the directory after the login process.



In both cases, if you have more questions related to your home directory, please consult your local system administrator.

The display path shown within the „Profile“ window is just a suggestion and can be overridden. If you want to store your „Profiles“ on one another location, you have to type in the new path, just replacing the displayed one.

More information related to Profile, such as, the structure or how to modify them via an editor can be found in the Configuration Section within the User's Guide (see Annex D).

After you have specify the location where your Profiles shall be stored, the Assistant is nearly ready to prepare your personalized Grasshopper environment onto your system. This installation can be activated by pressing the „Next“ window. If you like to return back to the previous window just push the „Previous“ button.

If you want to create a new path for your personal profiles, the checkbox right beneath the text field, where you have entered your new path specification, must be selected. Otherwise you will receive a error dialog window telling you that the profile path is corrupted.



Once the „Next“ is applied, the „Task“ window (see Figure 15) will appear on the screen. No particular functionality is provided by this window, it gives you a visual description of what the Assistant is going to prepare as well as the progress of the profile installation process itself.

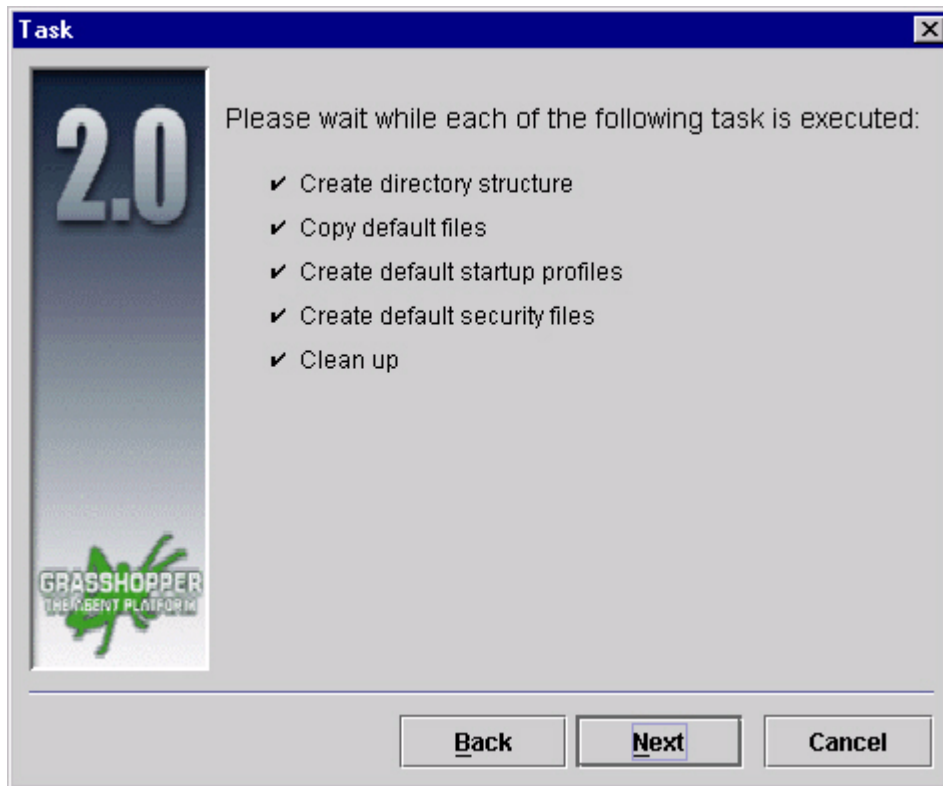


Figure 15: Task Progress Window

By applying the „Next“-button, you are ready to start your first Grasshopper application.

3.1.2 Selecting an Application to Be Launched

After you have setup your profile, you have to tell the Assistant whether you want to start an Agency or a Region Registry. The Assistant will support you in this respect by providing a „Choice“ window (see Figure 16) from where you can select either one of them to be launched.

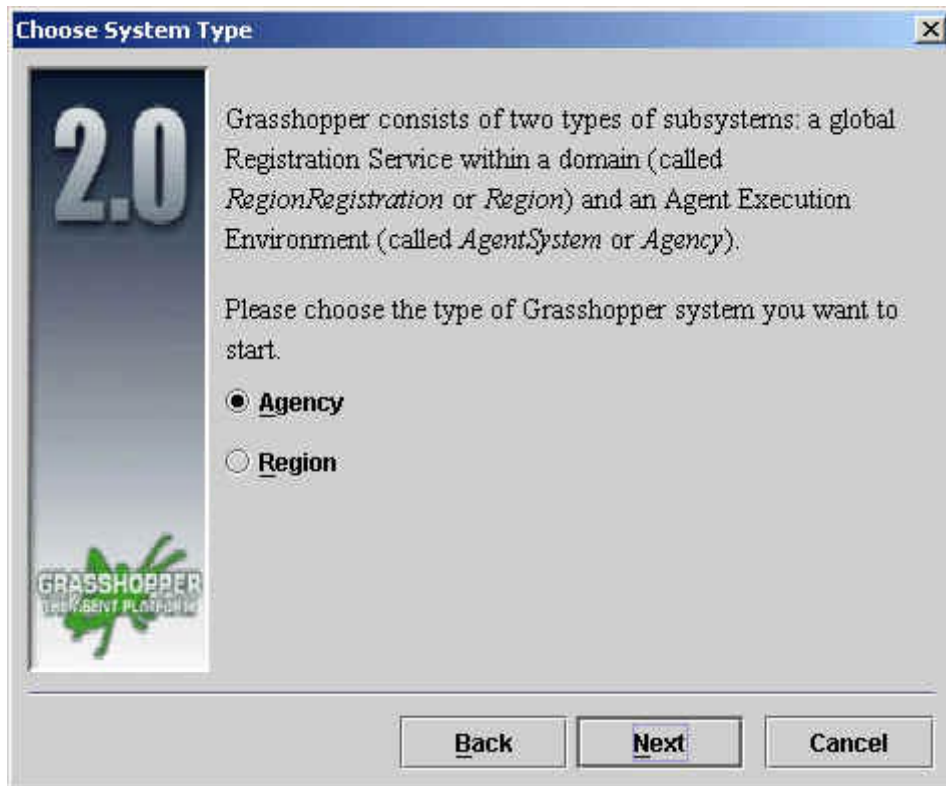


Figure 16: Grasshopper Application Choice Window

3.1.3 Selecting Profile to Be Used

Common to both Grasshopper applications, you have to specify the profile to be used by the Assistant for activating your selected application. The following „Selection“ window (see Figure 17) will be displayed on the screen once you have pressed „Next“.

As Shown in the Figure 17, you have three alternatives to select from. The first one, always selected as a default, is to use a default profile. The second enables you to load an existence one while the third alternative allows you create a new profile.

If you are asking where the Assistant finds your profile, remember you have set previously a profile path (see Section 3.1.1) in the „Profile“ window. This specified path is then used by the Assistant to find existence and to store your new profiles.





A more precise information related to profile setting possibilities can be found in Chapter 6.1.

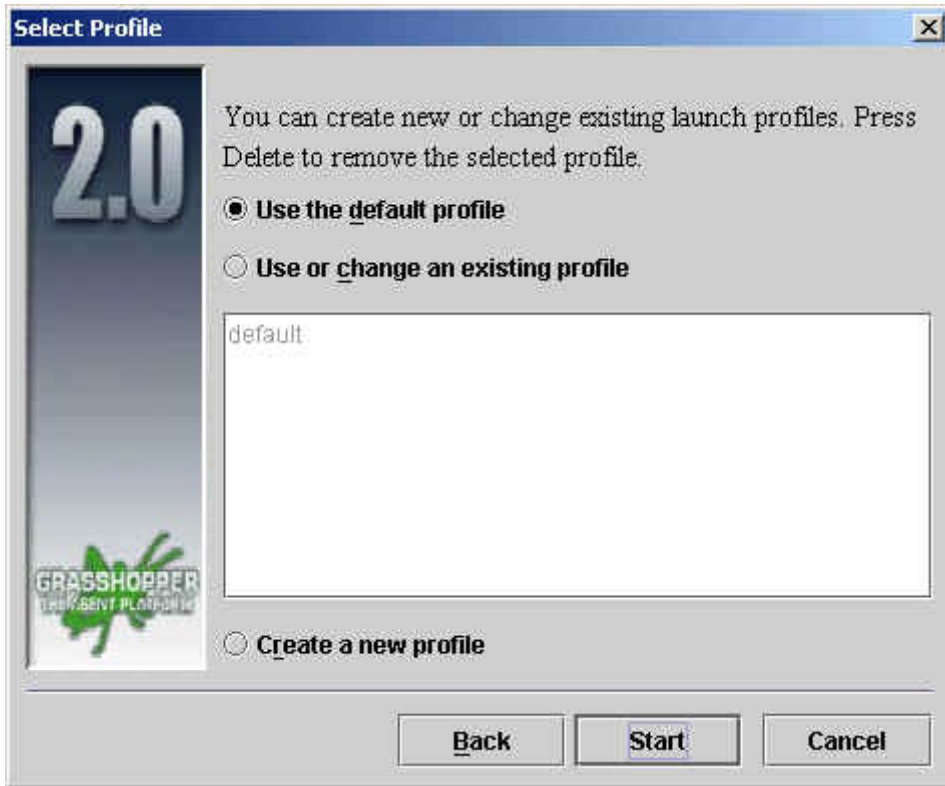


Figure 17: Profile Selection Window

Selecting to Create a New Profile

For example, if you select the last alternative, you can create a new profile. For this purpose, a profile configuration window as shown in Figure 18 is provided to you. Since you want to create a new profile, no particular settings are preset.

More details on how to set up a profile will be presented in Annex D, where the configuration possibilities of Grasshopper are described.

After you have setting-up a new profile, the Assistant will start your selected application.

For example, Figure 21 shows the graphical user interface of an Agency.

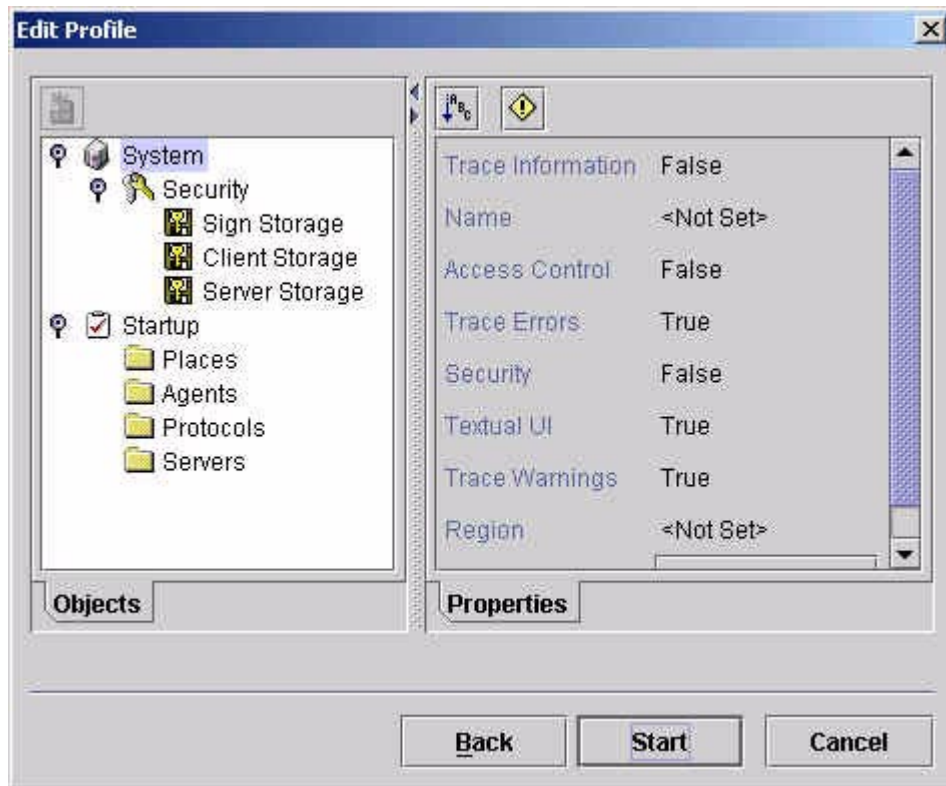


Figure 18: Profile Configuration Window

Selecting to Change an Existing Profile

Since this is the first time you have launched the Assistant, you will likely only find a default profile to select from. You have to create new profiles via the „New profile“ selection box in order to see other profiles than the default one.

If you select this option, the existing profiles will become selectable.

In order to select, you have to click the displayed name of a profile, which will then be set off. By concluding the selection, you have to press the „Next“-button.

Thereafter, a profile configuration window (see Figure 18), which contained the information of the selected profile, is shown to you.

Note, once a profile is set off, it can also be removed by pressing the „Delete“ button on your keyboard.

Once you have modified your profile, the Assistant will launch your selected application.



Selecting Default Profile

A default profile setting is shipped with Grasshopper and has been stored during the installation process (see previous Chapter 2: Installation) onto your system.

If you choose the default profile, the Assistant will launch an application that will have a graphical user interface as well as a text-based command-line interface, the so-called „Tui“.

3.1.4 Launching an Application

Once you have specified which profile to be used, the Assistant is going to activate either an Agency or a Region Registry depending on your selection (see Section 3.1.3).

Launching a Region Registry

If you have selected the Region Registry to be activated by the Assistant, the following window (see Figure 19) shall be seen on the screen. Since no Agency has been started so far, i.e., no Agency has been registered with the Region, the graphical user interface does not display any entries.

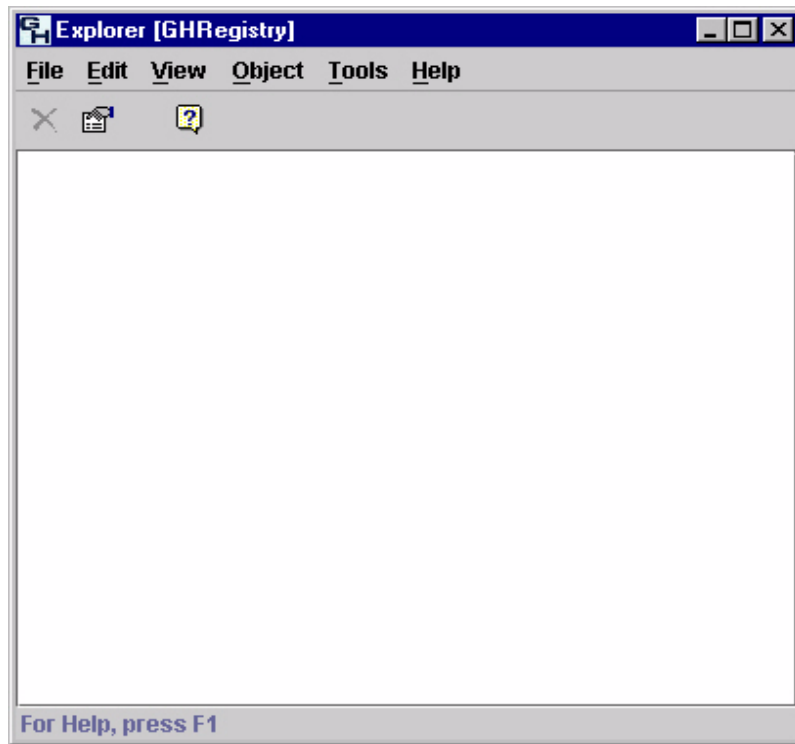


Figure 19: Region Registry Window

Also, a corresponding textual command-line interface is started. Please refer to Annex B for more information.

The message displayed is telling you that you have started a Region without any security support. More details on security can be found in Chapter 5.



```
D:\WINNT\system32\java.exe
Grasshopper Agent System V2.0 (C) 1998-99 by IKU++
10:55:31:375 e RegionRegistration: Failed to create security context, IAIK not found!
Region Text Console (C) 1999 by IKU++
Type 'help [command]' for more information
>
```

Figure 20: Region Textual Interface

Launching an Agent System

For example, if you have selected to launch an Agent System and to use the default profile, the following Agent System window (see Figure 21) will appear to you onto a Microsoft Window NT system. Depending on the profile and the system you used, the window may look differently.

A usage description of the Agent System window is presented in Chapter 5. Since no Agents or Places has been created so far, no particular entries is displayed within the graphical user interface.

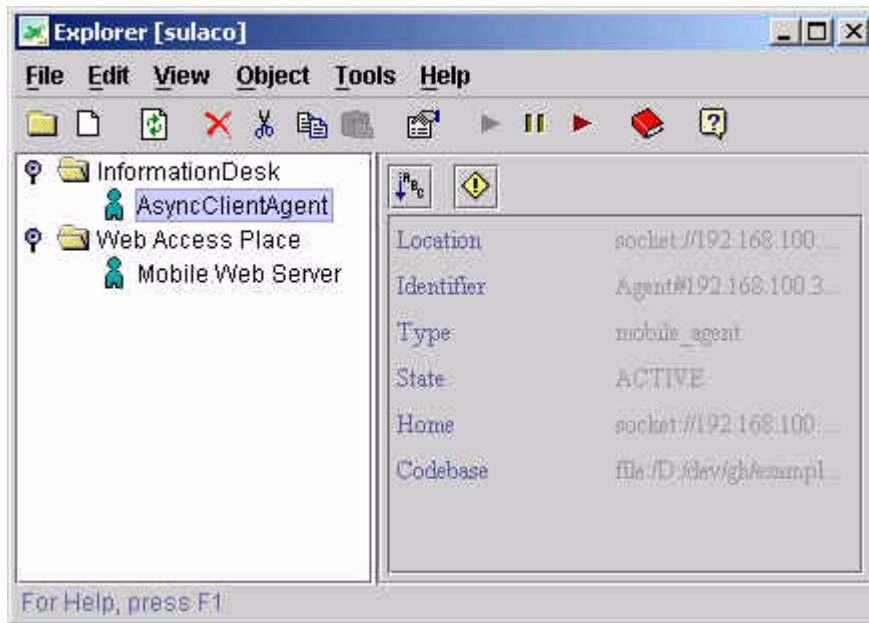


Figure 21: Agent System Window

The command-shell, from where you have typed in the command: „Grasshopper“, will -for an MS Windows system- look like as shown in Figure 22.

The usage of the textual command-line interface of an Agency it presented within Annex B. Please read this chapter for more details.



Figure 22: Agent System's Textual Interface Window

3.2 Launching an Application via Command-line

Besides the possibility of activating an Agency or Region Registry via the Assistant, Grasshopper also provide command-line options, which can be passed to the start-up program „Grasshopper“ that enables you to directly start an Agency or a Region Registry.

Launching the Assistant

How to launch an Assistant is already described in the previous Section. Please see Chapter 3.1, this will help you to find out information related to the Assistant.

Launching the Agency

If you want to launch an Agency directly, you have to pass the „a“ option to the „Grasshopper“ program.

For example, the following command-line will start-up an Agency with a graphical user interface as well as the textual command-line interface. With „-n“ option you can give the Agency a particular name.

```
Grasshopper a -n MyAgency -gui -tui
```

You can find the other applicable options to „Grasshopper“ either in Chapter 5, describing how to use an Agency or in the Annex A, where a comprehensive summary of all options are presented.

If you want that the agency is automatically registered to the Region Registry then the following command you have to use:

```
Grasshopper a -n -r socket://<ip>:7020/GHregistry.
```

Launching the Region Registry

In order to activate a Region Registry, you have to pass the „r“ option to the „Grasshopper“ program.

If you apply the following command as shown below within a command-line shell, a Region will be started that has a graphical as well as a textual command-line interface. The same as with an Agency, the option „-n“ is used to specify the name to be used to identify this application, in this case the name of the Region Registry.

```
Grasshopper r -n MyRegion -gui -tui
```

More information related options for starting a Region Registry via command-line can be found either in Chapter 6 or within the Annex A.



3.3 From Here ...

Now that you have learned the basics and the different way of starting-up a Grasshopper application, i.e., the Assistant, Agency or Region Registry.

4 Grasshopper Window System

In this chapter, you learn how to work with the Grasshopper graphical user interfaces. In order to get most out of Grasshopper, it is recommend that you are familiar with all the screen elements you'll be working with. This chapter helps you into taking your first steps toward using Grasshopper.

The following topics will be discussed within this chapter:

- How can you start and exit an Agent System or Region: This will take a look at starting an Agent System as well as a Region, using the command line and existing them with the command line or menu commands.
- Getting acquainted: You will learn to navigate through the graphical components of Grasshopper and get to use to their features.
- Using different View: Another purpose of this chapter is to provide you enough information so that you are able to display different views of Grasshopper and move around in the graphical components to fulfil your tasks.

4.1 Starting and Exiting Grasshopper

When you install Grasshopper, the setup program places will not place any icons on the screen nor put an entry into the Start menu (Windows). It only stores the Grasshopper program files either into the default directory or into the one you have specified during the installation process. More details on the installation topic can be seen in Chapter 2.2.

To start a Grasshopper application, you have these possibilities already described in the previous Chapter 3:

- Using the Assistant.
- Using the command-line.

The most convenient and fastest way to start an application is to use the command-line just by directly invoking the one you want.

4.1.1 Starting a Region Registry

In order to activate a Region Registry, you have to pass the „r“ option to the

„Grasshopper“ program.



```
Grasshopper r -n MyRegion -gui -tui
```

If you apply the following command as shown below within a command-line shell, a Region will be started that has a graphical as well as a textual command-line interface. The same as with an Agency, the option „-n“ is used to specify the name to be used to identify this application, in this case the name of the Region Registry.

4.1.2 Starting an Agent System



```
Grasshopper a -r socket://<IP>:7020/GHRegistry.
```

The argument *a* tells the Grasshopper to launch an Agent System that connects to the Region Registry with the address: `<IP>:7020/GHRegistry`.

You can find the other applicable options to *Grasshopper* either in Chapter 5, describing how to use an Agent System or in the Annex A, where a comprehensive summary of all options are presented.



There is one thing you have to keep in mind. An Agent System can only be registered to a Region Registry if it is up and running. Hence, you have to start the Region Registry first, otherwise your Agent System will throw an Exception, telling you that no Region Registry can be contacted.

4.1.3 Exiting an Agent System

Depending on the arguments you passed in to the Grasshopper start-up routine, i.e., the option *-tui* or *-gui*, there will be a command-line console and/or a graphical user interface, where you can enter your inputs.

Within the command-line console, you just have to type in *exit* to end the session. The exit command will clean up the system, i.e., removing and graphical user interfaces, terminating running Agents, and cleaning up all used resource of the Agent system.

Another way of ending the usage of Grasshopper is provided by the graphical user interface. The menu bar provides the exit function. To access the exit you have to go to: *File->exit* (see *Figure 23*).

Only for the graphical element, both the Windows and Solaris system provide a window system which enables you to terminate applications in general. Please refer to the corresponding descriptions. This feature can also be used.



Figure 23: Exit via Graphical User Interface

Once applied, a dialog box (see Figure 24) will be displayed which will ask you whether to *close* or *exit* the graphical use interface.



Figure 24: Exit Dialog Box

Don't be confused with the *close* command, which is also provided in the console and graphical element. This command will close the access point to the Agent System, but which will remain active. You are just losing the monitoring and controlling possibilities.



For example, if you use *close* in the graphical element, only the graphical user interface will be closed and not the Agent System.

4.1.4 Exiting a Region Registry

To terminate the Region Registry you can follow the same instruction as described for exiting an Agent System. Both the *exit* in the console or in the graphical element in the menu bar are provided.

4.2 Exploring Grasshopper Windows

You have probably realized that there are only slight differences in the view of the Agent System and the Region Registry window. Both are structuring in the same way. Many graphical components were reused in order to increase the recognition value.

Once you have started either an Agent System or a Region Registry, you will see the Grasshopper window. At the top you will see the *menu bar* and a *tool bar* below the menu bar. The center of the window is divided into two sections. The one section on the left will display the information entities in a *tree view*. The right section will show the properties of the information entities. The *property view* arrange the information in a table form. The *status bar* is visible at the bottom of the screen. The following Figure 25 shows the different graphical components of a Grasshopper window.

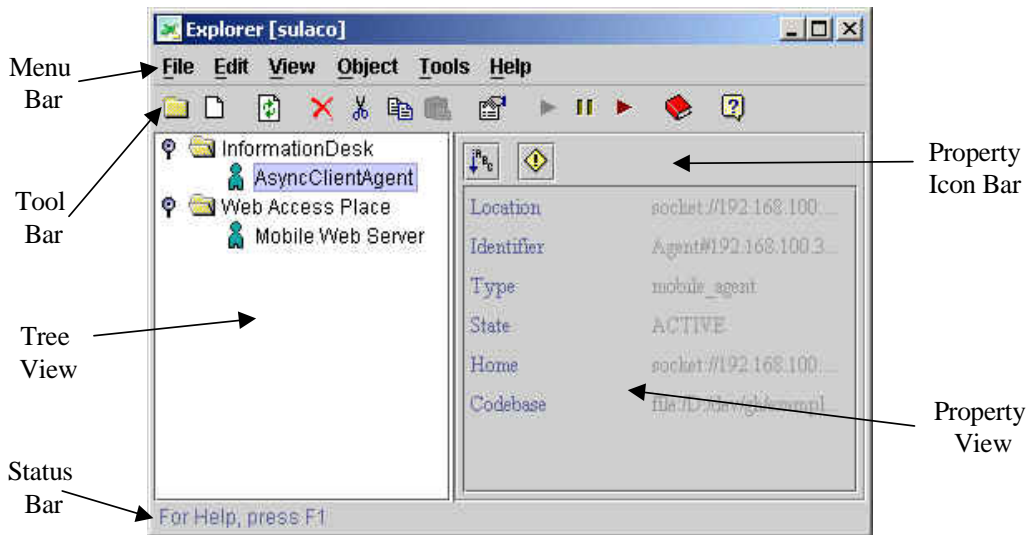


Figure 25: The Graphical Elements of Grasshopper Window

Menu Bar

The menu bar is very similar to menu bars of other applications, such as, Emacs or Microsoft Office products. The commands of the menu bar are defined and described in detail in later sections, as the functions they are perform are discussed (see Chapter 4.3 for Agent System and Chapter 4.4 for Region Registry).

Tool Bar

Appearing below the menu bar is the tool bar that contains buttons you can activate with the mouse to provide short-cut access to frequently used menu choices or special functions, i.e., the buttons differs whether you have an Agent System window or a Region Registry window.

The detail description will be provided as you encounter them by working thru this chapter.

Tree View

The tree view shows the information entities of Grasshopper. These entities are Agent System, Places, and Agents; Their relationship are reflected by a tree structure. For more complete descriptions of the tree view will be provided in connection with Agent System and Region Registry in later sections.

Property View

On the right side of the tree view, the property of the selected entity is displayed. Each entity is a holder of a set of properties, which will be presented at this view. A total list of properties of each entity is described in Chapter 6.1.1 for Agent System, Chapter 6.1.4 for Places, and Chapter 6.1.5 for Agent specifics.

The usable functions related to this view are going to be explained in the following next two sections.

Status Bar

The status bar is located at the bottom of a Grasshopper window. It displays advisory messages or descriptions of particular functions. Whenever you move a mouse pointer over an area within the Grasshopper windows which a particular function is associated, the status bar will display it.

For example, if you go into the *Menu bar->File->Exit* with your mouse pointer, the status bar will show the message: *Close all windows in this application and exit.*

Since no more functions are associated with the Status Bar, there will be no further description in the remaining sections.

4.3 Agent System Features

The Agent System window is the main entrance point for you to control and monitor the execution environment for agents and obviously your agent-based application.

The following Figure 26 shows an Agent System window.

4.3.1 The Menu Bar Interfaces

The Menu bar of the Agent System is shown in the following. Additional description of each menu will be provided in the next chapter of the User's

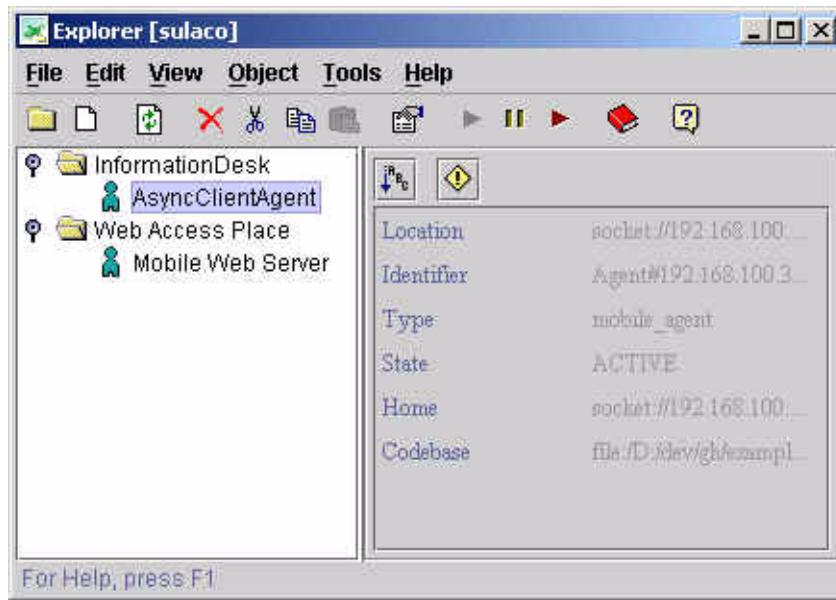


Figure 26: The Agent System Window

Guide. The Figure 27 shows the *File, Edit, and View*; while Figure 29 covers the menu entries: *Objects, Tools, and Help* of the menu bar items.

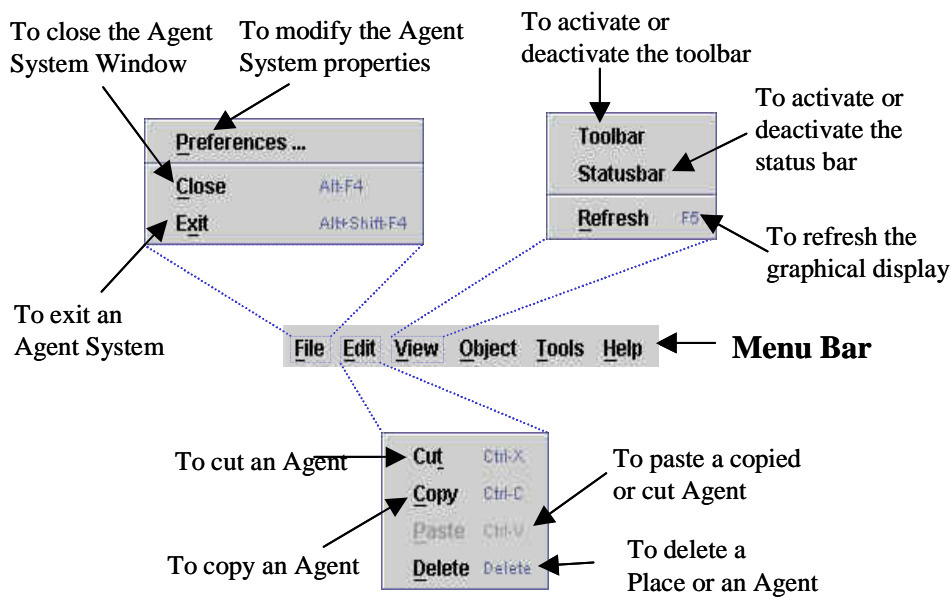


Figure 27: The Menu Bar Part 1

File

The File entry collects general commands applicable to the Agent System.

- The **Preference** will display a dialog window, where you will be able to customize the preferences (Properties) of the Agent System. A detail description of this issue is provided in Chapter 6.2. The Preferences window as shown in Figure 28 is modal, i.e., it will block all other windows.

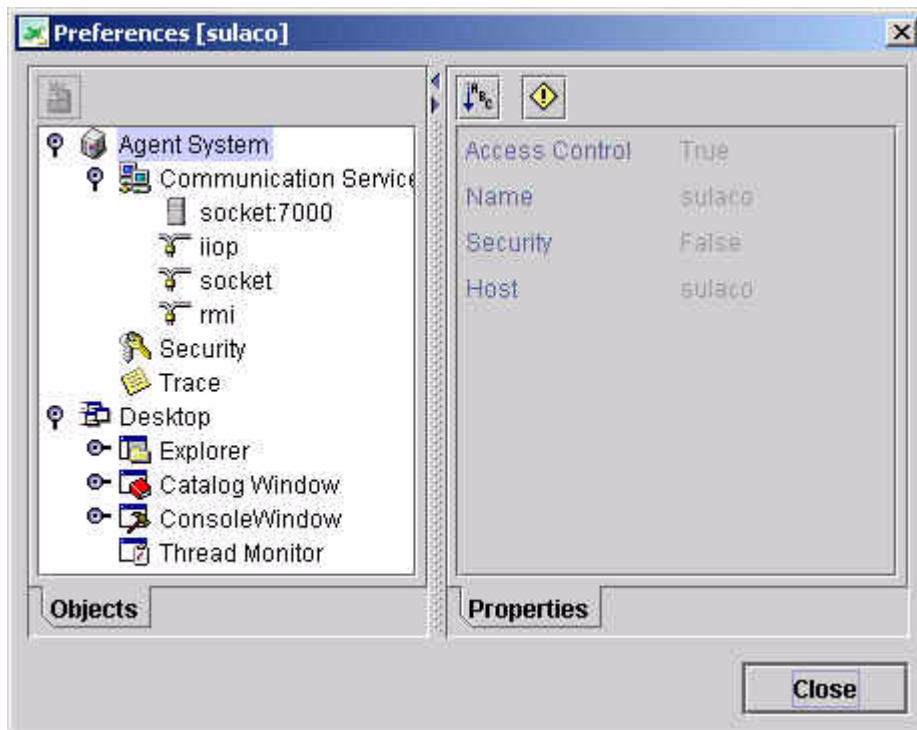


Figure 28: The Preferences Dialog Window

- The **Close** command will only hide all the graphical interface of the agent system. The agent system will still be available, though. Once this is selected, a dialog box will be displayed asking for confirmation, just for sure.
- The **Exit** command will terminate the Agent System. A dialog box will be displayed, asking for confirmation in order to avoid an activation by mistake.

Edit

The Edit entry contains commands to work with objects, such as, Place and Agent within a local Agent System.

- The *Cut* command is only applicable to an Agent. Thus, the menu entry

will only then be visible if you have selected an Agent within the Tree View. This command enables you to move an Agent from one Place to another within the same Agent System. You have to notice that the *Cut* is not really the command for agent migration, though. For this, the *Move* command of the *Object* entry shall be used.

In order to paste the Agent to another Place, you have to use the *Paste* command.

- The *Copy* enables you to copy an Agent to another Place within an Agent System. The actual copy of the Agent will be realized with the *Paste* command.
- The *Paste* command is the one to recreate an Agent. This entry will only be selectable once you have applied either the *Cut* or *Copy* command.
- The *Delete* command removes the selected entry within the Tree View. The selection can be either a Place or an Agent. Once deleted, the used resource will be freed.

View

The Grasshopper Agent System provides additional views which make optional information available or enables a user a faster access to key commands of Grasshopper. The purpose of the view is to allow the user to show or to hide these additional windows.

- The *Toolbar* provides short cut to key commands of the Agent System. Each icon represents a particular short cut as shown within Figure 42. The usage and functional description of each of them can be found within the intended sections of the User's Guide.



The activation of the toolbar follows an exclusive pattern, i.e., if the toolbar is already shown then the next time you press on the *Toolbar* command will hide it. The other way around behavior is given, if the Toolbar is hide.

- The *Status Bar* displays error message, warnings, or additional information, such as, help information. The *Status Bar* command enables you either to have such information shown or not. The same as the *Toolbar* command, it is also working in an exclusive mode. For example, if the status bar is displayed at the bottom of the Agent System Window, the pressing of the *Status Bar* command will hide it.
- The *Refresh* command will update the view of Agent System Window.

A „Please Wait“ message will be shown within the Tree View for the period of the refreshing process.

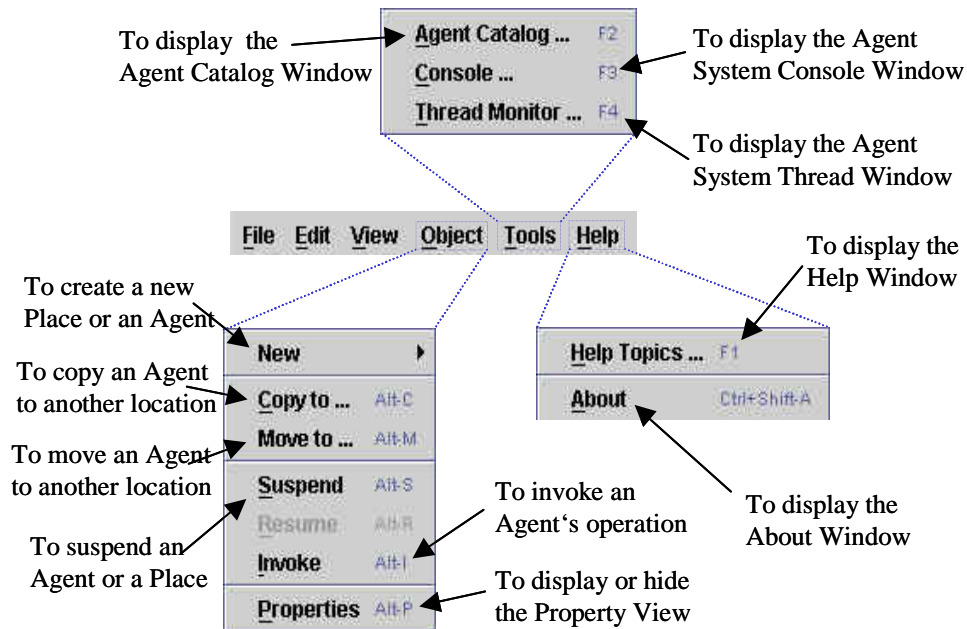


Figure 29: The Menu Bar Part II

Object

The Object menu entry collects actions applicable to Places and Agents. These are in particular:

- *New*: The *New* command enables you to create either a new Place or an Agent. For creating a Place, you have to specify the name and a description of the purpose of this place. Grasshopper provides for this a dialog box as shown Figure 30. For creating Agents, a specific agent creation

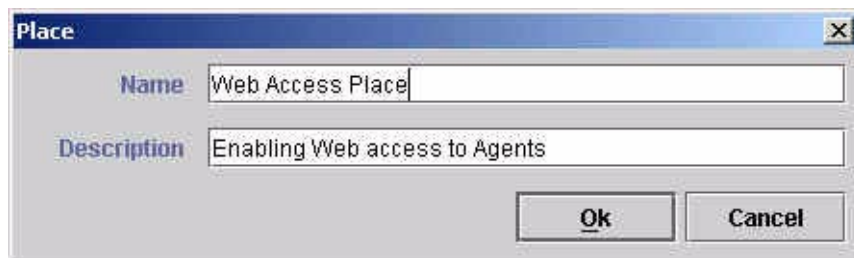


Figure 30: The Place creation dialog box.

dialog box is supported by Grasshopper. Associated with the creation,

you have to provide some information. These are the full-qualified Java classname, a possible codebase, a place name, and start-up arguments for the new agent instance. The gives an example, how such information may look like. More detail can be found in the Chapter 5.1.

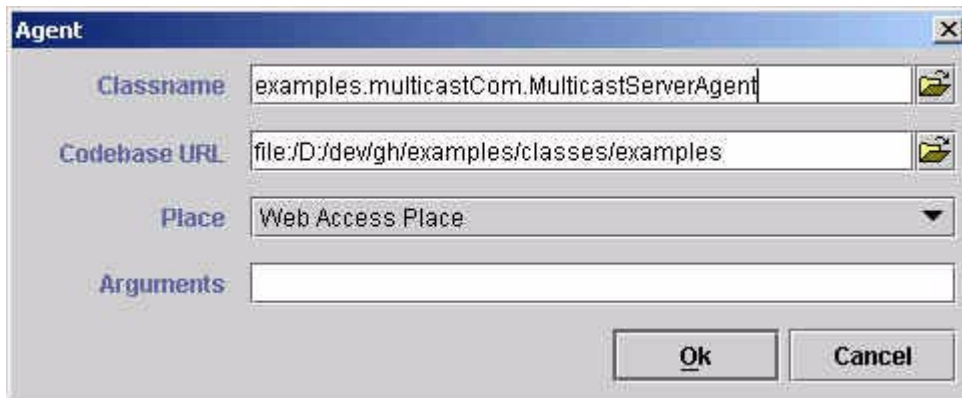


Figure 31: The Agent Creation Dialog Box.

- *Copy to:* The *Copy to* command enables you to create a mirror of a running Agent to another location, which is not bound only to the local one. For doing so, you have to provide a valid location address, which has to be entered within a dialog box as shown. The Copy to command can



Figure 32: The Location Specification Dialog Box

only applied to agents. Thus will only be activated within the menu entry after you have selected a Agent in the Tree View.

- *Move to:* The *Move to* command give you the possibility to request an Agent to migrate to another Location. For the specification of the location the above (Figure 32) dialog box is shown at the display.
- *Suspend:* The *Suspend* command enables you to stop an Agent's operation. This will only be usable once you have selected a not suspended Agent within the Tree View. Otherwise, the command is deactivated.
- *Resume:* The *Resume* command will resume a suspended to resume its operation. This command will only be selected if you have chosen an

Agent within the Tree View and this Agent is suspended.

- *Invoke*: The *Invoke* command executes the Agent's operation, i.e., the Agent's action-method. This command is only applicable to Agents which are not suspended. The selection of this command is once possible after you have selected an Agent within the Tree View.
- *Properties*: The Properties commands shows or hide the Property View and the Property Bar of the Agent System's Window.

Tools

The Tool entry includes a set of additional tools for helping you in creating Agents and providing a graphical information and status window. Along with these two and for system monitoring purposes, a thread monitoring view is made available. This tools shows all created threads of an Agent System.

- *Agent Catalog*: The Agent Catalog command will display a new window that enables you to create a book of frequently used Agents. This allows you to collect your favorite set of Agents and easily to create them in an one click manner.

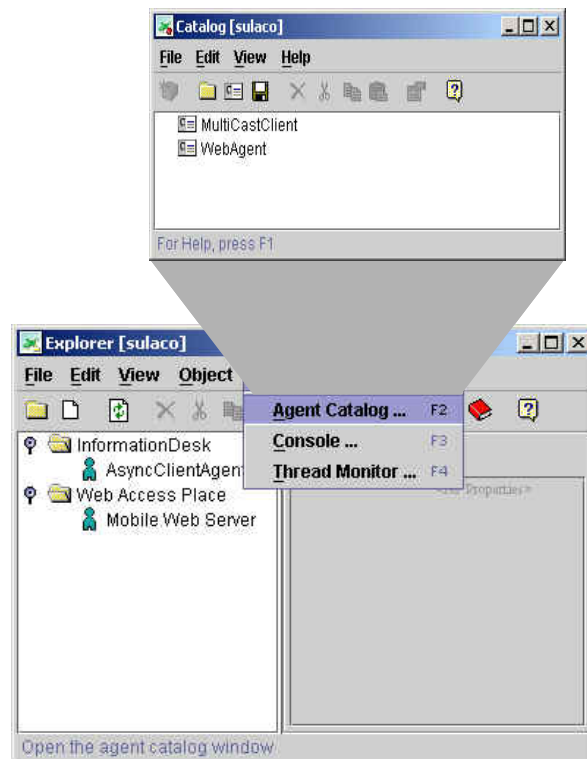


Figure 33: Agent Catalog Window

The *Agent Catalog* window has almost the same structure as the Agent System Window, except the missing Property view. The detail descrip-

tion of the Catalog can be found in Section 4.3.1.1.

- *Console*: The *Console* command will pop-up a new window, which will output system or agent messages. The structure of the window follows the same pattern as the Agent System Window. A description of this is given within Section 4.3.1.2.
- *Thread Monitor*: The *Thread Monitor* command provides a means to display the list of allocated threads of your Agent System. The following describes the meaning of the elements of the Thread Monitor Window. The here described thread structure is adopted from the Java

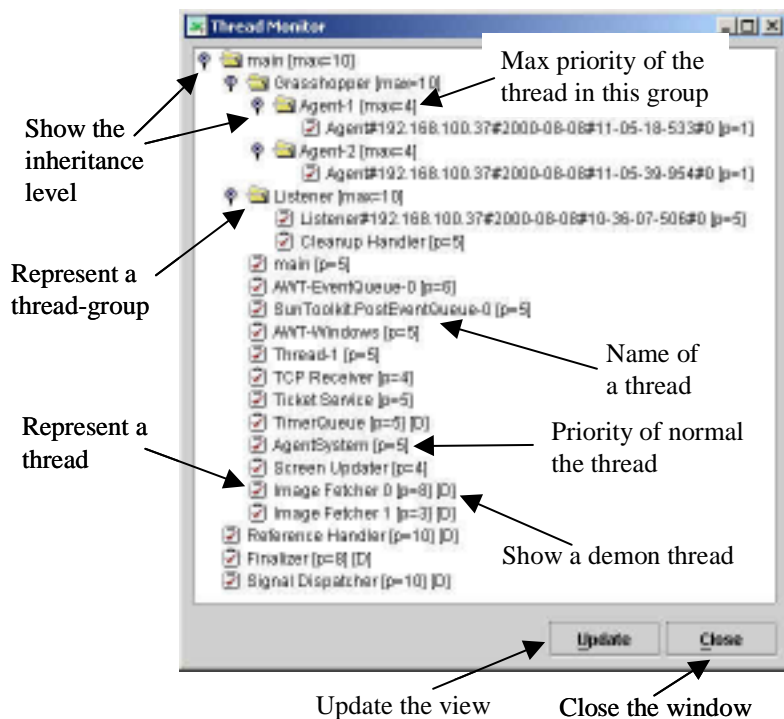


Figure 34: The Thread Monitoring Window

thread model. More details on threads can be found in the tutorial provided by SUN Microsystem under the URL:

<http://java.sun.com/docs/books/tutorial>.

Help

The Help entry of the Menu Bar offers access to the online help topics and the Grasshopper version number.

To access the help you have to chose the *Help Topics* command. Once selected, a new window will be displayed, where you can search on help topics.

In order to see the current version of your Grasshopper installation, you have

to access the *About* command. This will pop-up a new window, containing the version number.

4.3.1.1 Agent Catalog

The Agent Catalog Window, as shown in Figure 35, provides a means for you to build your favorite set of agents, combining it with a easy to create manner.

Following the same structure of an Agent System Window, the *menu bar* is at

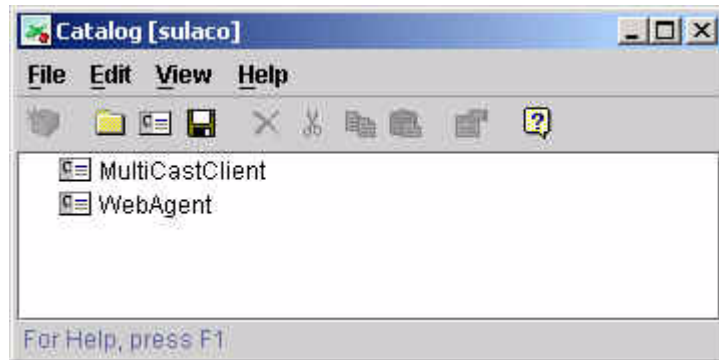


Figure 35: Agent Catalog Window

the top, the *toolbar* right below it and the *view part* is center part of the window. Last but not least, the *status bar* is at the bottom.

Menu Bar

The Menu Bar of the Agent Catalog contains the entries for *File*, *Edit*, *View*, and *Help*.

The *File* entries provide the commands:

- *Preferences*: This will start a new window, as shown in Figure 36, in which you can customize the appearing of the Agent Catalog Window..
- *Create*: The *Create* command will create the agent instance of the selected entry.
To create an agent, you select the agent entry in the View Part and go with the mouse pointer to this command and press your mouse button.
- *Save*: The *Save* command will save the setting of the agent catalog to the default preference file. This preference file contains your personal properties of your Grasshopper. This file is located at your user home directory.

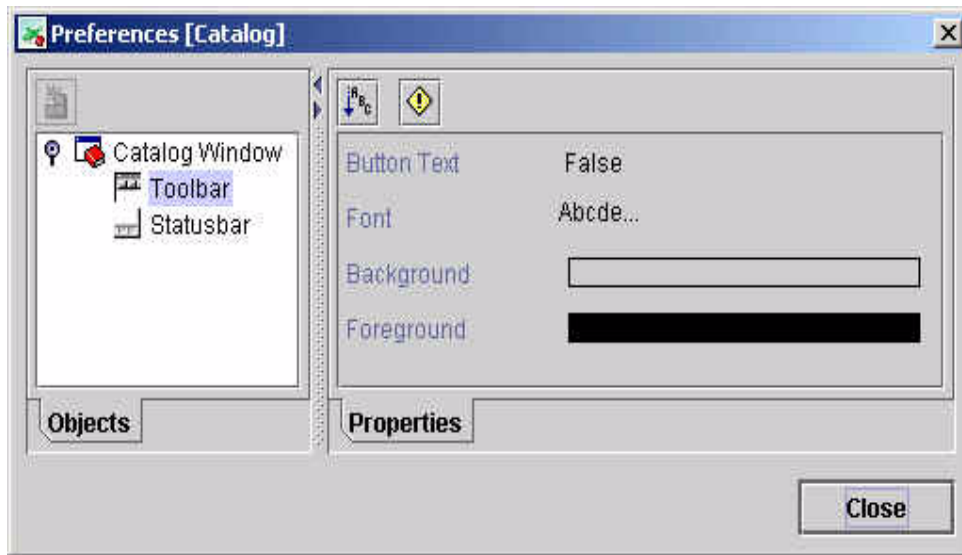


Figure 36: Preference Window of the Agent Catalog



For Windows NT, this preference file can be normally found under `c:/winnt/profiles/<username>/grasshopper`. If you are using Windows 2000, this file is located under `c:/Documents and Settings/<username>/grasshopper`.

There is one exception for the location of the file, it is reported that if you are logged in as root and for some circumstances, this file is either not created or save to another location.



For Unix system, the preference file can be found under `/user/local/<username>`

- *Save As:* You you wish to save the Agent Catalog's file to another location, you can use this command and just a new location specification for the file.

Remember, you this file will not be automatically loaded within the next start up of Grasshopper. You have to explicit use the *Load* command.

- *Load:* This command enables you to load another Agent Catalog's file either than the default one.
- *Close:* This will close the Agent Catalog and will update the current loaded preference file.

The *Edit* command has the same structure and provides the same functions as the *Edit* of the Agent System' Menu Bar. These commands are: *Cut*, *Copy*, *Paste*, and *Delete*, which can apply to folder and agent entries. The only additional is the *Properties* entries, which enables you to view and modify the

property value of a folder or an agent entry.

The *View* command provides functions to activate or deactivate the *Toolbar* or *Status Bar* of the Agent Catalog window.

The last provided command, the *Help*, provides you help information on Agent Catalog.

Toolbar

The Toolbar contains buttons you activate with the mouse to provide a short-cut access to frequently used Catalog menu choices or special functions.

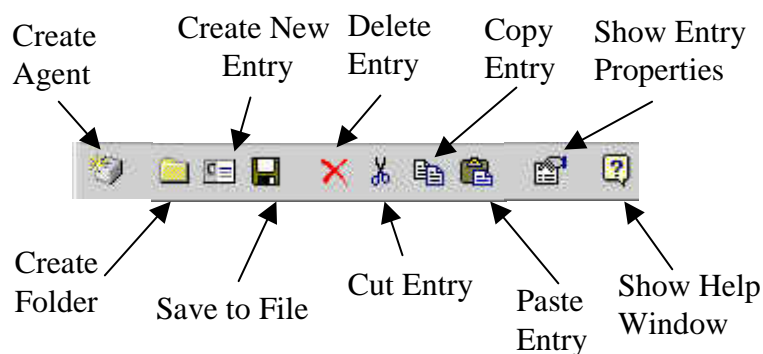


Figure 37: Toolbar of the Agent Catalog

- *Create Agent*: This icon will be available once you have selected an agent entry. By moving the mouse to and by pressing the icons, an agent will be created based on the entry specification.
- *Create Folder*: This icon will only be activated if you have not selected an agent entry. It enables you to create a new folder, helping you to organized your entries. Once applied, a dialog window will appear and ask you of the name of the folder. Along the name, you can also provide some description information of the folder.
- *Create New Entry*: This icon will be deactivated if you have selected an agent entry. Once pressed, an agent entry dialog window will be shown. An entry requires the property types as shown Figure 38.
- *Save to File*: By using this icon you can save all your setting into the preference file (see above), which enables the preservation of your inputs. This preference file will be automatically loaded by the catalog next time you launch the Agent Catalog Window.
- *Delete Entry*: The icon will be available after you have selected an

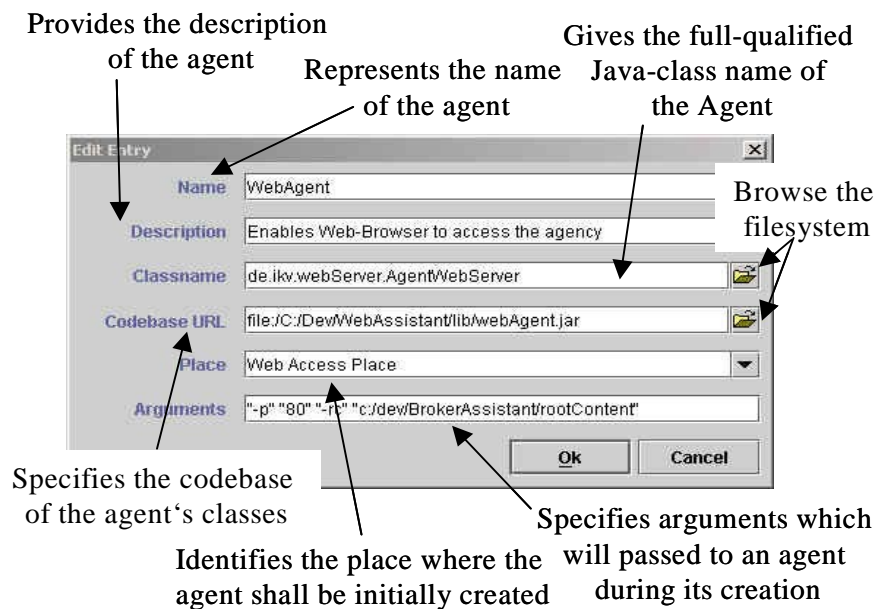


Figure 38: Agent Entry Window

agent entry. The selected agent entry will then be removed from the catalog after you have confirmed the deletion.

- *Cut Entry*: This icon enables you cut an entry from the entry view and to put it, for example, into another folder. This will only apply onto agent entries, thus you have to select in order to use.
- *Copy Entry*: This will make a copy of the selected agent entry, enabling you to paste it to another folder. This will only usable if you have selected an agent entry.
- *Paste Entry*: This icon will only appear if you have previously copied an agent entry. In order to use it, you have to deselect the copied agent entry.
- *Show Entry Properties*: This icon will show you either the properties of the selected folder or agent entry. If you want to modify the property then you have to use this icon.
- *Show Help Window*: During your usage, if you have a problem then you might find the answer in the online-help just by pressing this icon.

View Part

This view shows you all the agent entries contained within your preference file.

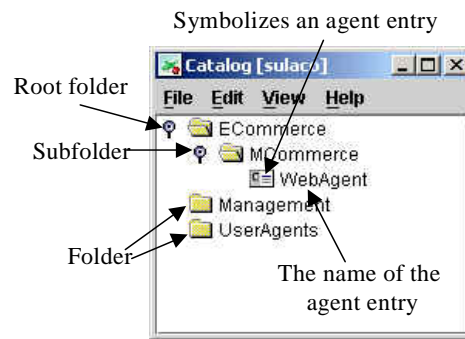


Figure 39: Agent Catalog View part

Status Bar

The same as the Agent System main window, the status bar shows you additional messages or name of the function you have selected.

Tips

There is also another fast way to activate most of the described commands. Just by pressing the left button of your mouse after you have selected an entry within the Tree View (see Section 4.3.3).



Another convenient feature provided by the Agent Catalog is the automatic creation of an Catalog entry for an existing Agent. For this, you just have to select an Agent icon shown within the *Tree View* of the Agent System and move it to the Catalog window.



4.3.1.2 Console

The *Console* Window provides a convenient way to view Agent System or Agent written out messages.

The window is structured as the other Grasshopper window components, with a menu bar, toolbar, view part, and a status bar as reflected in Figure 40.

Menu Bar

The Menu Bar contains the entries: *File*, *Edit*, *View*, and *Help*.

The *File* component includes the commands:

- *Preferences*: This will start a new window, as shown in Figure 36, in which you can customize the appearing of the *Console* Window.
- *Close*: This will close the *Console* window.

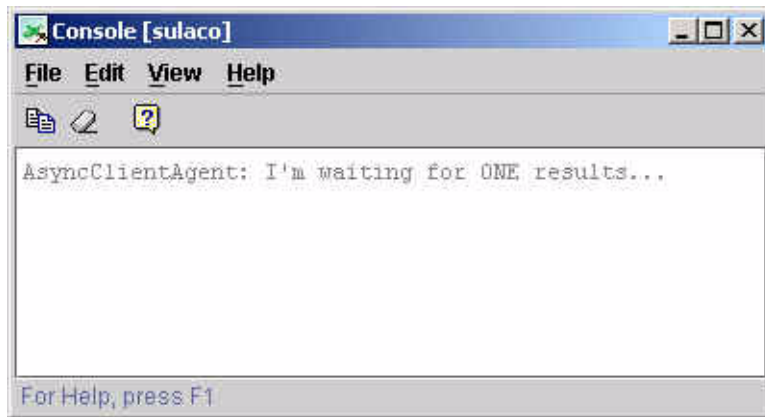


Figure 40: Agent Console Window

The *Edit* command provides a *Copy*, allowing you to copy a message from the *View* part into, for example, an Email or editor. The other function, the *Clear*, will clean the *View* part.

The remaining commands, the *View* and *Help*, provide the same structure and functions as described in the previous section of the Catalog.

Toolbar

The *Console's* *Toolbar* provides the short-cuts as shown in *Figure 41*.

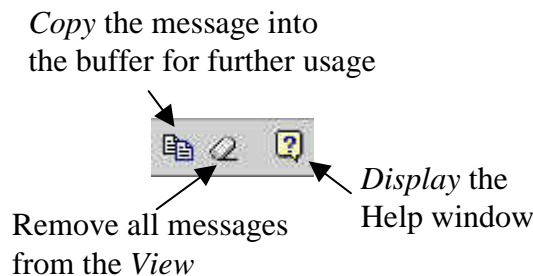


Figure 41: Toolbar of the Console

View part

The center of the *Console* window represents the *View* part, in which the messages are displayed.

A context menu can be activated just by pressing the left button of the mouse. The context menu includes the command: to *Copy* messages and to *Clear* the view.

Status Bar

The same as the Agent System main window, the status bar shows you additional messages or name of the function you have selected.

It can be activated or deactivated by using the provided command within *View* located at the *Toolbar*.

4.3.2 The Toolbar

The Toolbar collects the frequent used commands as shown in Figure 42. Some of the icons are state dependent, i.e., they are only available if certain conditions are fulfilled.

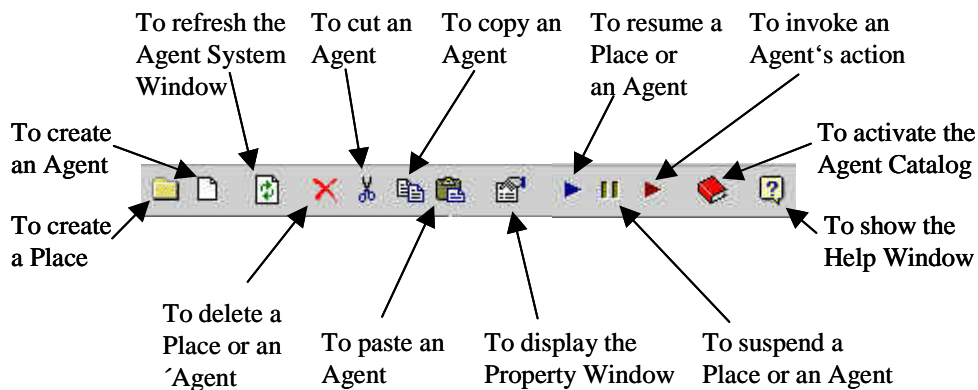


Figure 42: Toolbar of the Agent System Window

These short-cuts are available, starting from the left most to the right most:

- *Create Place*: This will create a new place, which will be displayed in the *Tree View*. You can only create new places at the same structural level as the *InformationDesk*, the default place of an Agent System. Sub-Places are not supported. Once applied, a new dialog window will appear and ask you the name, a must attribute, of the place and optionally you can provide along a description of it.
- *Create Agent*: The next icon enables you to create an Agent. For doing so, you have to provide some creation information to the Agent System. These pieces of information will be requested from you via a dialog window as shown Figure 43.

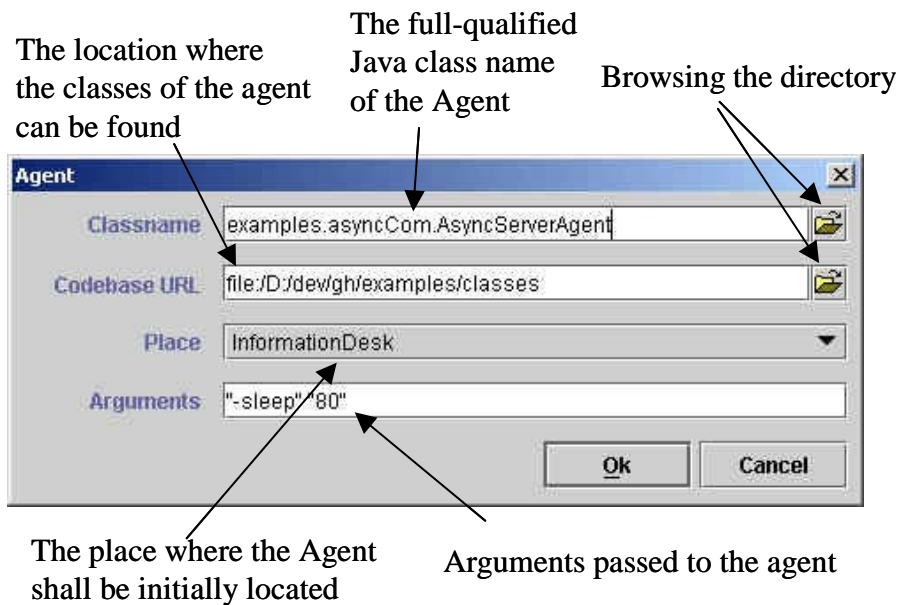


Figure 43: Agent Creation Dialog Window

- *Refresh View*: Even though, the Agent System Window has the ability to refresh the view automatically, it is sometimes required that you have to update the window manually. Then, this icon you have to press. The refreshing process will become apparent within the *Tree View*, where „Please wait...“ message is displayed.
- *Delete Selection*: You can remove a place or agent with this icon. Thus, it will only be available once you had selected either a Place or an Agent within the *Tree View*. You have to remember that by deleting a place you will also erase all sub components of it. There is also one exception: You are not allowed to remove the default place, the *InformationDesk*.



Once you have removed an entry, e.g., an agent system entry, the Region Registry does not provide any procedure for re-register the information. The re-registration has to be done by the instance itself (see Annex B.2.7 for a registration).

- *Cut Agent*: If you like to copy an Agent from one place to another within an Agent System then this command can be used. Do not become confused with the *Move* command as described earlier in this section. The *Cut* only allows you to reorganize the location of Agents within the local Agent System, while the *Move* enables you actually to deploy an Agent to another location. This function only applies to an Agent.

- *Copy Agent*: If you like to duplicate an Agent within a local Agent System then this icon will provide the right function for you. The Copy can only be applied to Agents.
Once used, a duplicate is made and cached within the Agent System. The copy will only become into effect with the *Paste* command.
If you want to copy an Agent to another Agent System then you have to use *Copy to* command located at the *Object* within the *Menu Bar*.
- *Paste Agent*: The Paste-icon will only be available if you have previously either has *Cut* or *Copy* an Agent.
- *Display Property Window*: This will either activate or deactivate the *Property Window* next to the *Tree View*. More details can be found Section 4.3.4.
- *Resume Selection*: If you like to continuing the computing of an Agent or all Agents within a Place, which were previous suspended, then you have to press this icon. This icon will only be applicable if *Suspend* were executed previously.
An Agent can not be directly resumed if the suspension were cause by the suspension of a Place, e.g., the Agent was a member of the suspended place. It can only be resumed if the Place is resumed.
- *Suspend Selection*: You can stop temporarily the computing of an Agent just be suspending it. All Agents within a Place can be suspended if you just suspend that Place. Remember those Agents can only be resumed if you resume the Place.
- *Invoke Agent's Action*: To start the Agent's task, you can explicitly use the *Invoke* command represented by this icon. This icon will only be visible if you have selected an Agent, which is not suspended, within the *Tree View*. Further, it is not applicable to Places.
- *Show Agent Catalog*: This icon will display the *Agent Catalog Window*. More detail description of it can be found in Section 4.3.1.1.
- *Show Help*: This icon will display the *Help* window.

4.3.3 The Tree View

The *Tree View* represents the main information of an Agent System. It is the center part of the Agent System Window. The following Figure 44 shows all the graphical element shown within the *TreeView*.

Additional, within the *Tree View* and once you have selected an entry, either a Place or an Agent, you can activate the *Context* menu just by using the left but-

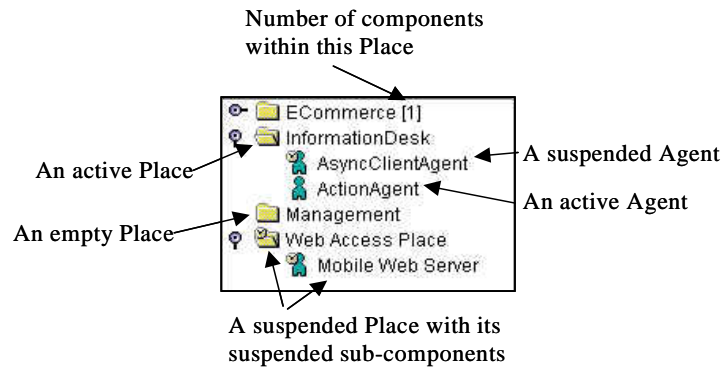


Figure 44: Graphical Elements of the Tree View

ton of your mouse. The *Context* menu provides a list of applicable functions of the selected entry.

For example, the following shows the list of functions related to an Agent. No additional functions are provided than the one as described within the *Menu Bar* or *Toolbar* of the *Agent System Window*. More details on the provided functions can be thus found at the corresponding section of this chapter.

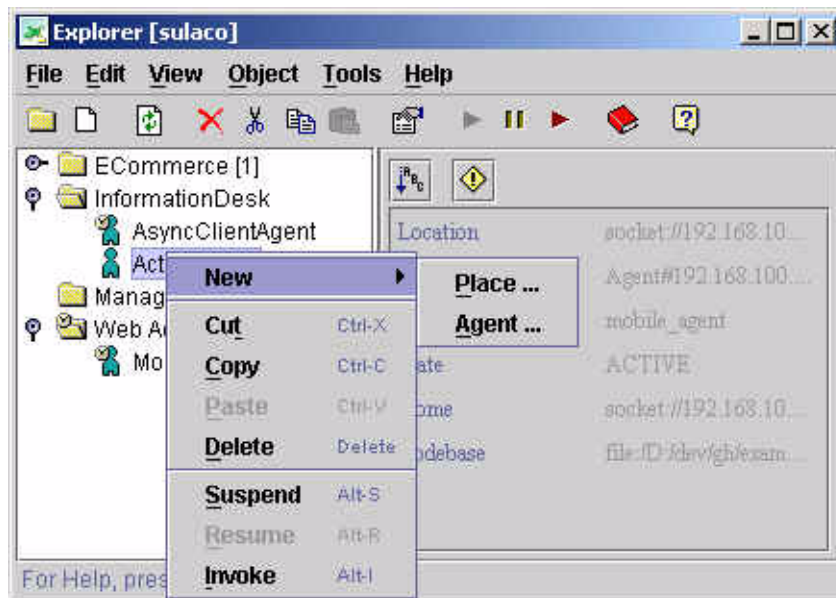


Figure 45: Context Menu within the Tree View

4.3.4 The Property View

The Property View is an optional window, which can be activated or deactivated. There are three possible way to do this. The first way is to use the *Properties* command provided within the *Context* menu. Another alternative is to press on the property icons of the *Toolbar* as described in Section 4.3.2. The last one is provided within the *Object* entry of the *Menu Bar* as described in Section 4.3.1.

At the top of the Property View, there is a toolbar providing two short-cuts. The first one enables you to sort the properties of the selected entry, either a Place or an Agent. The second one will show expert properties that are not frequently used.

Below the toolbar, the property types are displayed at the left side while the associated value are at the right side of the view. Depending on whether you have selected a Place or an Agent, a different set of properties are shown. The list of available properties are explained in the Programmer's Guide.

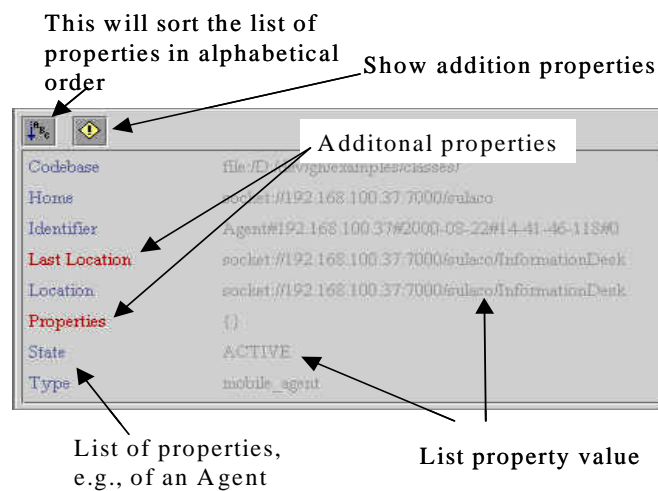


Figure 46: Property View of the Agent System Window

4.4 Region Registry Features

The central registration unit of the Grasshopper Agents and Agent Systems is the Region Registry. Each of them will send their new state information to this instance once it had been changed.

The graphical representation of the Region Registry are based on the same

components as described for the Agent System. Thus, do not wonder if you have a déjà vu during the reading of the Region Registry Window system.

The Figure 47 shows a Region Registry with no registration information.

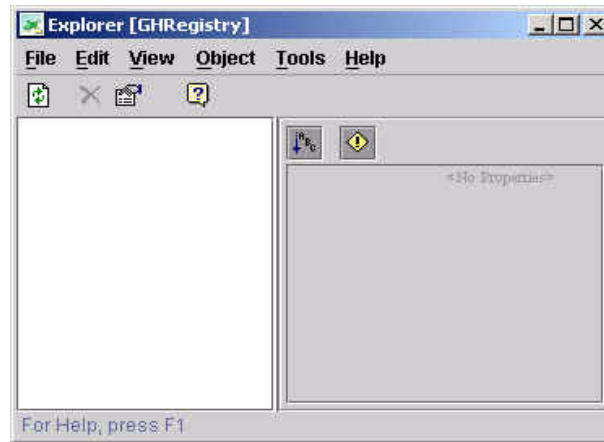


Figure 47: Region Registry Main Window

4.4.1 The Menu Bar

The Menu Bar of the Region Registry contains the commands as shown within Figure 48. It is structured in the same way as the Menu Bar of the Agent System. Thus, it provides the following categories of commands: *File*, *Edit*, *View*, *Object*, *Tools*, and *Help*.

These commands will be described in particular in this section.

File

- The *Preference* command enables you set up Region Registry' properties. The setting is provided in a dialog window, where you will be able to customize the preferences of the Region Registry. The what and how to set properties will be provided in Chapter 6.2. The Preferences window as shown in Figure 28 is modal, i.e., it will block all other windows.
- The *Close* command will hide the Region Registry windows. The actual process though will be still active, though. For example, you can reach the Region Registry via the textual interface at the command line level.
- The *Exit* will close down the Region Registry.

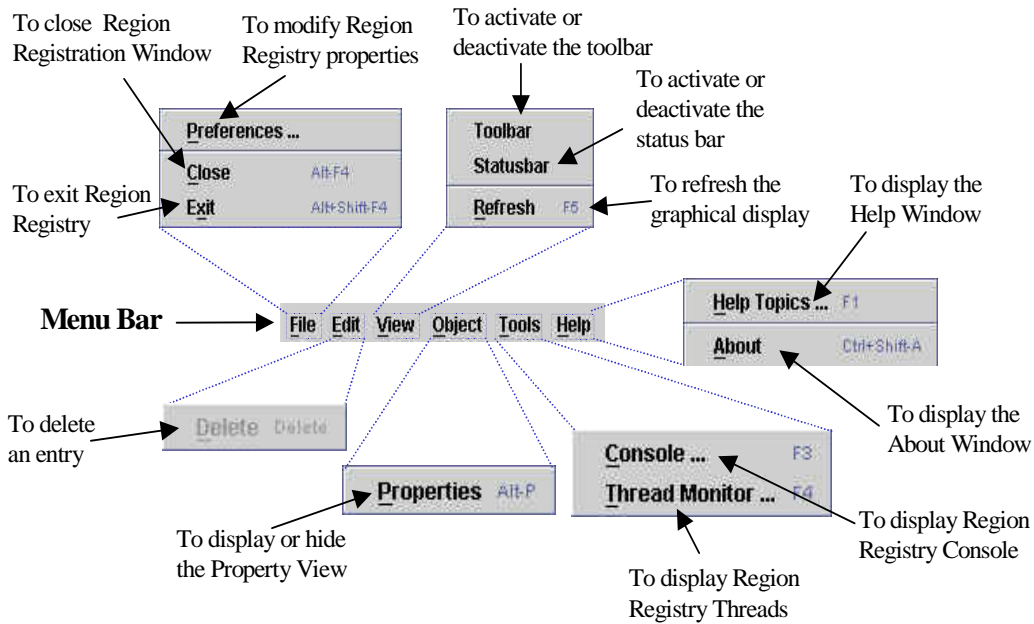


Figure 48: All Entries of the Menu Bar

Edit

The only command you can reach within this category is the deletion (*Delete*) of registration entry. Remember, just by removing an entry doesn't mean that the actual instances are removed.

View

The View contains the same commands as described in Section 4.3.1 as the Agent System View category was explained.

- The *Toolbar* provides short cut to key commands of the Region Registry. Each icon as shown below represents a particular short-cut



The activation of the toolbar follows an exclusive pattern, i.e., if the toolbar is already shown then the next time you press on the *Toolbar* command will hide it. The other way around behavior is given, if the *Toolbar* is hide.

The description of each icon is provided Section 4.4.2.

- The *Status Bar* displays error message, warnings, or additional information, such as, help information. The *Status Bar* command enables you either to have such information shown or not. The same as the *Toolbar* command, it is also working in an exclusive mode. For example, if the

status bar is displayed at the bottom of the Agent System Window, the pressing of the *Status Bar* command will hide it.

- The *Refresh* command will update the view of Region Registry Window.

Object

The *Properties* command allows you to show or hide the *Property View* of Region Registry window. See Section 4.4.4 for more details.

Tools

If you open the Tools, these two commands are available:

- *Console*: The *Console* command will pop-up a new window, which will output system or agent messages. The structure of the window follows the same pattern as the Agent System Window. A description of this is given within Section 4.3.1.2.
- *Thread Monitor*: The *Thread Monitor* command provides a means to display the list of allocated threads of your Region Registry. Since this window is the same as the one described within Agent System, you can find description of this window in Section 4.4.2.

Help


If you are looking for answers for your question regarding the usage of Region Registry, you may find it in the online help facility of Grasshopper. Having activated the *Help* command, you can search on help topics.

In order to see the current version of your Grasshopper installation, you have to access the *About* command. This will pop-up a new window, containing the version number.

4.4.2 The Toolbar

The *Toolbar* of the Region Registry contains less short cuts than the one of the Agent System.

The following four short-cuts, as displayed in ,are provided:

- Refresh : For refreshing the graphical view or updating the registration view, you have to press on this icon. The updating process will take some performance time which will be made known to you in the *Tree View*.

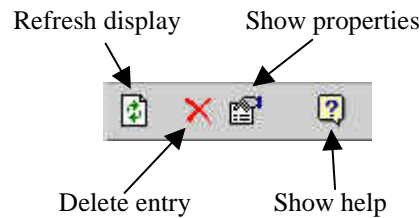





Figure 49: Toolbar of Region Registry

- Delete : This icon will only be available if you have selected an entry within the *Tree View*. It will turn its color from grey to red then. You can delete from an agent and place entry to a complete agent system entry. Remember, the sub entry will also be removed if you delete a super entry. For example, by removing an agent system entry all sub entries, such as, places and agents are also erased.

Once you have removed an entry, e.g., an agent system entry, the Region Registry does not provide any procedure for re-register the information. The re-registration has to be done by the instance itself (see Annex B.2.7 Thefor a registration).



- Properties : If you like to see the properties of each entry, you have to press on this icon. To hide this view, just press on this icon again. More details of the property view is provided in Section 4.4.4.
- Help : This will show the modal *Help* window.

4.4.3 The Tree View

The *Tree View* of the Region Registry has nearly the same structure and provided features as described for the one for the Agent System.

Since the Region Registry is the central registration unit, the display must have the ability to separate the different Agent System entry available within the network. Thus, this is the only diversion from the *Tree View* of the Agent System.

The different registration entries are shown in Figure 50.

Applicable actions related to entries within the *Tree View* can also be activated via the left button of your mouse. A context-sensitive menu bar will then be

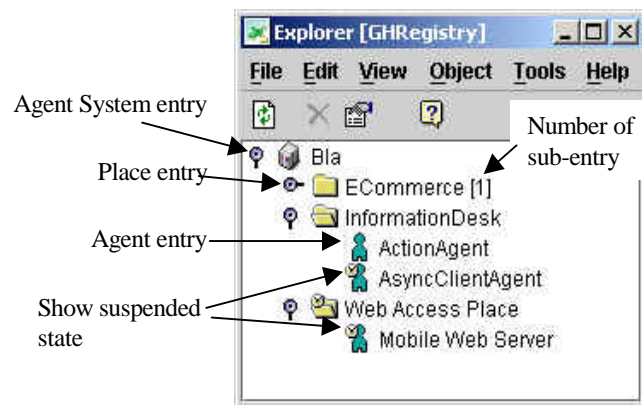


Figure 50: Region Registry Tree View

visible. The *Delete* and *Properties* command are provided. The behavior description of these two can be found in the previous section.

4.4.4 The Property View

The *Property View* of the Region Registry displays the properties associated with the selected entry. A change of the property value is not provided, since the Region Registry is just a reflection of the *Property View* of the Agent System one.

More details on this view can be found Section 4.3.4.

4.5 From Here ...

The various sections of this chapter has explained pretty much in detail the graphical user interfaces of Grasshopper and their functional behavior behind. You are now able to navigate thru the provided functions. This prepares you for the next chapter, which will describe the Agent life cycle aspect.

If you think you are ready to go into more details and internal stuffs then you can go directly to Chapter 6, where settings of Grasshopper will be explained.

5 Grasshopper Usage

What are you expecting in this chapter? In the following, a usage description of the Grasshopper's System's is given.

At first, after you had acquaint yourself with the usage of the graphical user interfaces you are ready to use the provided functions related to Agent, Place and Agent System. Therefore, you will find information about some usage scenrios, explaining the potential and characteristics of the each of them.

You will find the following topics in this this chapter:

- First, you will find a sequence of events to start-up your Grasshopper environment.
- After you has started-up the environment, your agents are ready to be launched. Section 5.2 will explain what functions you can apply onto agents.
- Agents runs always within a Place, enabling the grouping of agents either based of functional or structural matters. The provided features for managing places are described Section 5.3.
- The last section will describe some typical scenarios for using the Agent System's features. The scenario will include the life-cycle management of Agents, such as, starting an Agent System, monitoring and usage of Agents.

5.1 Starting-up Grasshopper Environment

5.2 Agent Life Cycle

Grasshopper provides a complete support of Agent life cycle, i.e., it provides functions from the creation, customization to removal Agents.

In detail, these are the following functions:

- Creation (Section 5.2.1)
- Removal (Section 5.2.2)
- Suspension (Section 5.2.3)
- Resumption (Section 5.2.4)

- Copying (Section 5.2.5)
- Moving (Section 5.2.6)
- Invoking Action (Section 5.2.7)

5.2.1 Creating an Agent

Grasshopper provides several alternative ways for you to create Agents. These possibilities are:

- Using the profile to specify Agents that will be automatically started during the start-up of an Agent System. For this you can either use the Assistant (see Chapter 6.1.5.1) to create a new or modify an existent profile or use an editor to add an entry into the profile by hand (see Chapter 6.1.7.2).
- launching the textual interface and types in at the command-line level the *Create* method. A detail description of this command can be found in Annex B.2.1. For example, to create an Agent with no codebase and paramter specifications shall look like this:

```
create agent examples.ActionAgent
```

- taking the graphical supports provided by the Agent System. These are in particular:
-

5.2.2 Removing an Agent

5.2.3 Suspending an Agent

5.2.4 Resuming an Agent

5.2.5 Copying an Agent

5.2.6 Moving an Agent

5.2.7 Invoking Agent's Action

5.3 Organizing your Agents using Places

5.4 Usage Examples

5.5 From Here ...

6 Customizing Grasshopper

Many possibilities for customizing Grasshopper with your personal touch is provided. This section will introduce all aspects to you.

Generally, there are four parts where you can configure Grasshopper. Three of them are provided by means of a visual component. The last one is more a hidden one, where you can modify directly some configuration files via a command-line or text editor.

The following aspects will be covered by this sections:

- The configuration of the Grasshopper applications prior to their first initialization. These configuration actions are associated with the name „Profile Management“ (See Chapter 6.1).
- The second and the third configuration possibility is related to the setting of preferences for Agent System and respectively for a Region Registry. These are described in Chapter 6.2
- The remaining parts then are dealing with setting related to the general environment of Grasshopper. Within this section you will find out many internal and advanced setting possibilities for those who want to be a power-user of Grasshopper.

6.1 Customizing Your Profile

As described in Chapter 3 where -among other things- you have learn just little about the profiling system of Grasshopper, this section will reflect on more detail about the start-up configuration possibilities for specifying your initial Grasshopper settings.

Please note that the content of the „Profile“-Window varies from the type of the Grasshopper system. The Region system allows you to set less properties than an Agent System.

The „Profile“-Window for an Agent System is shown in Figure 51. A „Profile“-Window is structured into three areas. One, the left side, displays the objects where you can selected from. Second, the right side, contains the attributes from the selected objects. Third, below these both panels, the navigation area is presented.

Grasshopper provides two types of properties, one for system related setting and second for start-up process of agent system or region registry.



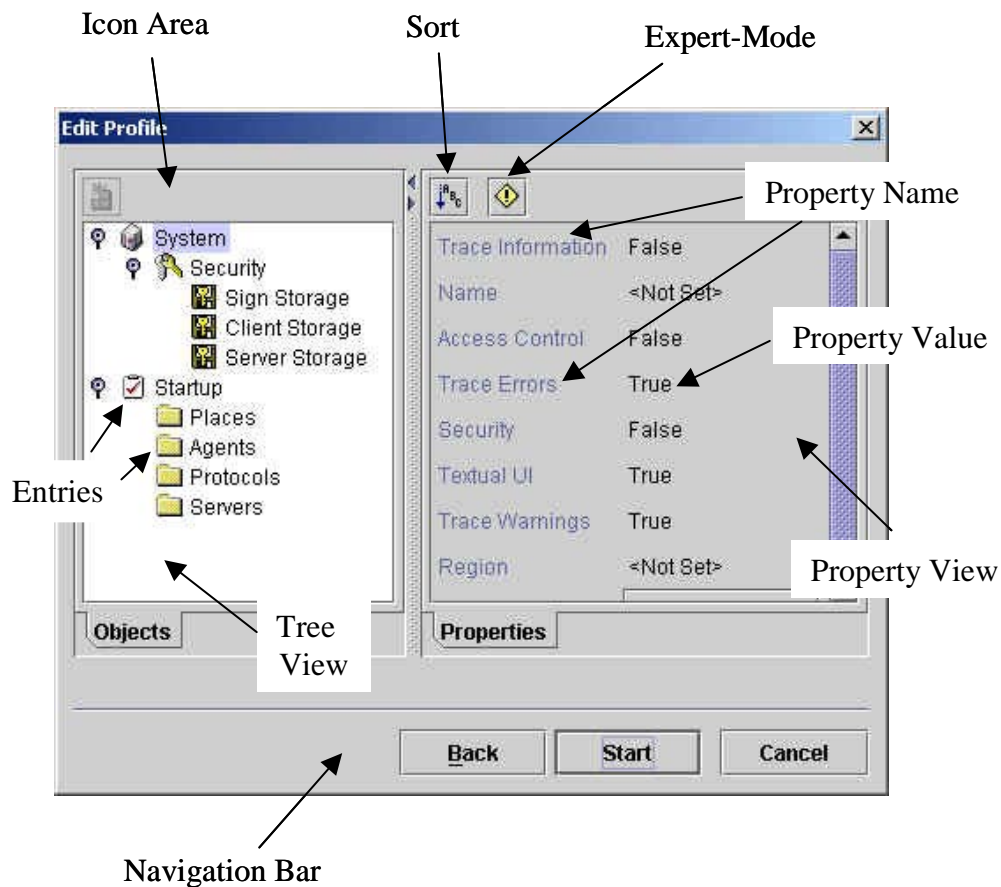


Figure 51: Profile-Configurator for Agent System



- **System:** If you wish to set some properties generally related to the Grasshopper system then you have to select this object. This entry is used to identify Grasshoppers' component which can be either an Agency or a Region. Property values of System such as tracing information or the activation of the textual user interface can be found. Also, the security setting, such as, the sign keys can be found here.

The following entries are all related to the start-up process:

- **Places:** You can also specify some places that shall be created initially. The Place object can only be selected by an Agent System.
- **Agents:** With Agent Properties, you can define Agents that will be created during the start-up of Grasshopper. This object can only be selected by an Agent System.
- **Protocols:** You can configure which communication protocol shall be supported by Grasshopper.

- Servers: A set of communication server can be declared and will be started.

Above the Property View, there are two icons on which you can click on. The meaning of each icon is:

- By pressing this , the displayed properties within the Property View will be sort alphabetically. The icon will then change its color into a deeper gray. Again, pressing it will put the properties into their initial order.
- Once pressed on , additional and not frequently used properties will be also shown. These properties are colored in red.

The related properties of each types will be described in more detail starting at the next following sections.

6.1.1 System Properties

The system entry is selected on default. The first table identifies the common properties applicable for both the Agent System and the Region Registry. There followed the typical properties for Agent System and Region are shown separately.

Common Properties

Associated with the System are the following common properties:

| Property Name | Purpose | Value Type |
|---------------|--|------------|
| Name | This identifies the name of the system (Agent System or Region Registry). The value can be any textual strings. For example: you can type in „MyFirstRegion“ in order to give your Region system this name. This name is then used for identification purposes. | String |

Table 6: Common System Properties

| Property Name | Purpose | Value Type |
|-------------------|--|------------|
| Textual UI | With this property you can define whether a textual user interface shall be started. A value set to true means with textual user interface support. | Boolean |
| Graphical UI | Same as the textual user interface, you can also specify that a graphical user interface shall be started. True means with while false without. | Boolean |
| Trace Information | With this property set to true you enable Grasshopper to print out trace information either on the textual interface or within the graphical console. The provided information is telling you about the momentary executed action within the system. Note, these pieces of information are not errors. | Boolean |
| Trace Format | This property allows you customize the output message of the trace information. | String |
| Trace Warning | In addition to trace information which are related to normal action, any warnings will be provided if this property is set to true. Otherwise no warning will be printed out. | Boolean |
| Trace Error | Once you have set this property to true, any errors closely related to Grasshopper will be shown either on the textual or graphical interface. | Boolean |

Table 6: Common System Properties

| Property Name | Purpose | Value Type |
|---------------|--|------------|
| Log | <p>If you want to log all console outputs then this property has to be set to true. You must note, in order to log also security relevant information, you must set the <i>Audit File</i> property to true.</p> <p>All information will be stored in the file as specified with the <i>Log File</i> property.</p> <p>This property is viewable after you have pressed the Expert-button.</p> | Boolean |
| Log File | <p>This specify the file to be used by Grasshopper to store all the console outputs.</p> <p>This property must represent a valid filename. The file will be either created or the existing one will be used.</p> <p>This Property is only visible once you have pressed the Expert-button.</p> | String |

Table 6: Common System Properties

| Property Name | Purpose | Value Type |
|---------------|--|------------|
| CORBA Args | Grasshopper can be used with ORBs from different vendors, such as Iona (OrbixWeb) or Sun (Java-IDL). If the ORB needs some parameters then you have to specify them with this property. For example, if you want to access SUN's CORBA Naming service from Grasshopper you have to pass the following parameter with this property to Grasshopper: <code>-ORBInitialHost <hostname></code> and <code>-ORBInitialPort <portnumber></code> . Please refer to the handbooks of the ORB vendor documentation for other parameters. This property will only be shown if you have pressed on the Expert-buttons. | String |
| Security | In order to enable the security service, you have to set this property to True. Otherwise, your Agent System or Region Registry remain insecure. | Boolean |

Table 6: Common System Properties

For the security setting of the system, there are further properties. The *Security* entry has two common properties as shown in Table 8.,

| Property Name | Purpose | Value Type |
|---------------|--|------------|
| Client Auth | If you like that the client always have to be authenticated then you have to set this property to TRUE. Otherwise no authentication of the client is forced. | Boolean |

Table 7: General Security Settings

| Property Name | Purpose | Value Type |
|---------------|---|------------|
| Server Auth | Same as the client, if an authentication of the server is required, then this must be set to TRUE. No authentication of the server is then otherwise. | Boolean |

Table 7: General Security Settings

Below the *Security* setting, there are two other security types of setting, which are addressed for the SSL communication among Agent Systems.

- **Client Storage:** Represents the key store of the client part of the SSL communication.
- **Server Storage:** Represents the key store of the server part of the SSL communication.

Associated to these two, there are the following four properties as shown in Table 8.

| Property Name | Purpose | Value Type |
|---------------|---|------------|
| Location | The value of this property point at the location where the key are stored. This location is named as KeyStore. | URL |
| Provider | This identifies the provider of the KeyStore. Since Grasshopper is using SUN security packages, the default provider name is „SUN“. | String |
| Password | This represents the password for the KeyStore | String |
| Type | The value of this property identifies the type of the KeyStore. For example, Grasshopper is using the Java KeyStore, thus the default will be set to „JKS“. | String |

Table 8: Common SSL Security Properties

Agent System Specific Properties

This sections contains Agent System specific properties.

| Property Name | Purpose | Value Type |
|----------------|--|------------|
| Access Control | You can activate the access control mechanism of Grasshopper. Once activated, the security manager will force the security policy as defined within the file referred by the <i>Policy File</i> property. Each requested action will be check against the access control policy. | Boolean |
| Policy File | This property point at the file that contains the security policy which the security manager has to follow. The value of this property is a string as for example: /home/usr/gh-user/myPolicy This property will only be shown when the Expert-button had been pressed. | String |
| Audit File | In contrast to the <i>Log</i> property, the <i>Audit File</i> property, once set to true, will log all security relevant information to a specified file. Such logged information can be used for security management purposes. The value of the property is a string and shall identify a file name, such as d:/users/gh-users/security.log The <i>Audit File</i> property can only be seen in the expert mode. | String |

Table 9: Agent Specific Properties

| Property Name | Purpose | Value Type |
|---------------|--|------------------------|
| Persistence | Grasshopper is equipped with a persistent object service, providing a common interfaces to the mechanisms used for retaining and managing the persistent state of objects. For example, an Agent's dynamic state is typically in memory and is not likely to exist for the whole lifetime. This state can be written out a file and be used to reconstruct the dynamic state. Once this property is set to true, all agents supporting this feature will be hold persistent. You should note that the agents must meet some requirements in order to use this feature. Please see the Programmer's Guide for more details. This property is an expert one, thus only visible when the Expert-button has been pressed. | Boolean |
| Region | The address used to contact a Region Registration system is set via this property. Please note that this property only applies to an Agent System, thus won't appear for a Region Registry setting. An example for a region address is such as socket://myHost:7025/ myRegion. | Grasshopper Address |

Table 9: Agent Specific Properties

For the Agent specific security setting, there is one additional property related to the *Security* entry (see Table 6). This entry point identifies the certificate to be used for signing agents.

| Property Name | Purpose | Value Type |
|---------------|---|------------|
| Sign Alias | This allows you to give an alias for the certificate to be used for the authentication process. | String |

Table 10: Agent Related Security Properties

Further and below *the Security* entry, there is an additional sub-entry, which is named *Sign Storage*. This property's purpose is to identify the key store to be used for securing agents and Agent Systems. Associated to the Sign Storage are the four properties as described in Table 8.

Region Specific Properties

The following shows Region specific properties:

| Property Name | Purpose | Value Type |
|---------------|--|------------|
| JNDI Password | The Region can work with a directory service as LDAP for storing information that must be globally available within a network boundary, such as within an Intranet. This property contains the password for the authentication process, authorizing a user as specified by <i>JNDI Username</i> property. This property will only be visible if the expert mode is activated. | String |
| JNDI Username | The name of the user shall be assigned to this property. This property will be displayed if the you have changed the display into the expert mode just by pressing the Expert-button. | String |


Table 11: Region Specific Properties

6.1.2 Protocol Properties

Grasshopper allows you to import your own communication protocol. Two properties are supported: one for the fully-qualified Java class name that represents the protocol implementation and the port on which the protocol will listen to.

6.1.2.1 Adding a New Protocol Entry

There are two possibilities to add your own implemented protocol to Grasshopper via the graphical user interface. The first thing you do is to select the folder icon. Thereafter, with the pointer directly over the icon, you use your second mouse button to activate a context menu, on which you can select „Add Protocol“. Once you have done this, a dialog as shown in Figure 52 will appear and in which you can add the classname and the default port properties.

Besides using the second mouse button, you can also use the icon  to add your protocol.

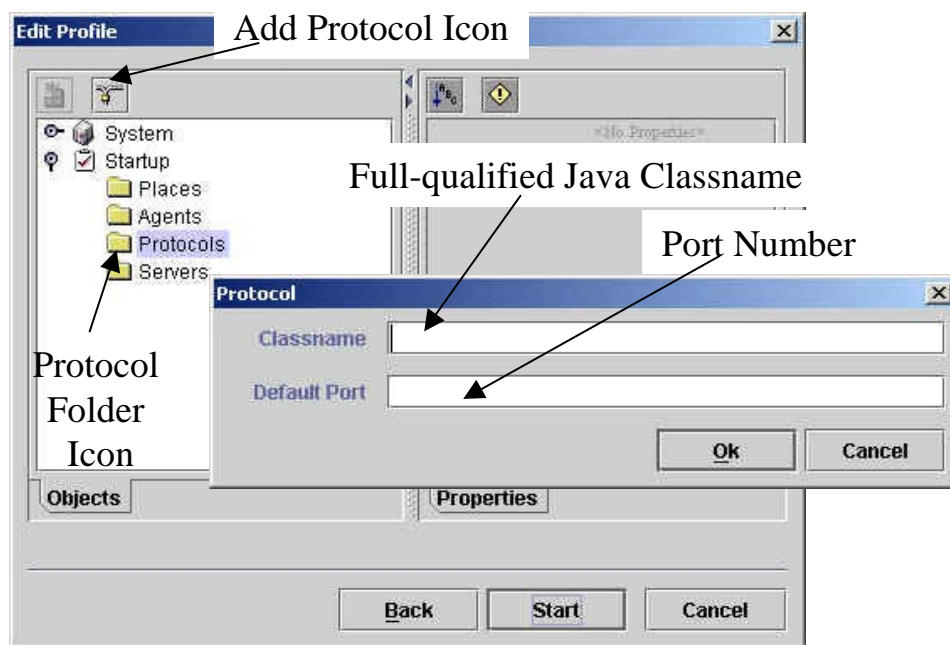


Figure 52: Adding new Protocol to Grasshopper

Once added, your own written protocol is known to the Grasshopper system and can be used by the communication service. In order to select, you have to associate a protocol with a server.

6.1.2.2 Removing a Protocol Entry

In order to remove a „Protocol“-entry you have to select the respective one beneath the „Protocol“-folder. After the selection, the entry shall be highlighted. The actual removal can be accomplished by using the „Removing“-Icon at the top in the „Icon“-area. The alternative is to use your second mouse button to activate the „Remove“-menu.

6.1.3 Server Properties


Communication among Grasshoppers' entities are realized by means of a Communication Service, which is an essential part of each Agent System. This service allows location transparent interactions among Agents, Agent System, Region, and non-agent-based components.

The communication service is implemented as a set of servers, where each server supports various protocols. Initially, Grasshopper supports IIOP, RMI, and Socket. Other protocols can also be added.

Within the profile setting, you can configure the Communication Service, i.e., which server shall be initially created, which protocol it shall support and on which port it shall listen.

6.1.3.1 Adding a New Server Entry

In order to add a new server to be started initially, you have two alternatives:

The first one, you can use the „Add-Server“ icon  as shown in Figure 53 or you can use your second mouse button to activate a menu by pointer directly over the „Server“ icon within the „Objects“ panel. Either way, a dialog box will then appear on which you can enter the server properties.

For adding a server, you need to select the protocol to be supported and on which port it shall listen to. If there is no reason, why the server has to listen to a specific port, you can enable Grasshopper to dynamically assign a port to the server in case the specified one is already occupied by another server. All these inputs will be requested by the „Server“ dialog box.

6.1.3.2 List Properties of the Server Entry

After you have created a new entry, you will see that a new icon is attached to the „Server Folder“ icon. This icon represents the server entry. On default, this entry is named followed by this pattern: „server- \langle number \rangle “. The number will be automatically increased by the number of created entries.

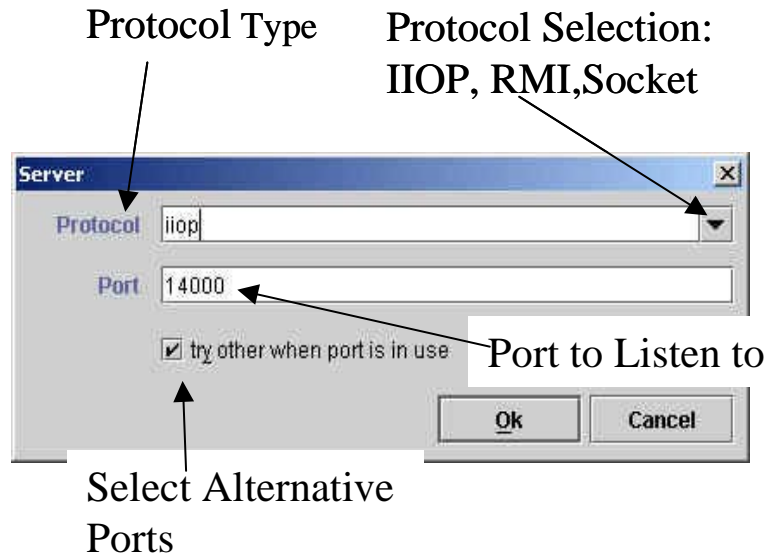


Figure 53: Adding Servers

Once you select this icon, a set of properties will be shown in the „Property“ panel. These properties are the following as depicted in table:


| Property Name | Purpose | Value Type |
|---------------|--|------------|
| Port | The <i>Port</i> property contains a value that tells the communication server on which port it shall listen to. | Integer |
| Name | The <i>Name</i> property is only for labeling purpose. Thus, is used as an Identifier and therefore, must be unique. | String |
| Description | If you want to provide some textual information related to the server, then you can put it right here, in this property. | String |
| Protocol | The <i>Protocol</i> property identifies the supported communication protocol of the server. | String |

| Property Name | Purpose | Value Type |
|----------------|---|------------|
| Host | This property shows you the name of the host on which the server is running. The value will be assigned by the Agent System. Therefore, this property is not settable. | None |
| Address | This property identifies the address of the server. This address is required if a client wants to communicate with this server. However, the value of this property will be assigned by the Agent System, after the server has been created. Thus, you are not able to set any value. This value will only be seen if the Expert-button is pressed. | None |
| Try Other Port | This property enables the server to listen to another port in case the specified one is already in use. The value TRUE will activate this feature. In case the port is occupied and this value is not set to TRUE, the server will not be started. This property will only be seen if you have pressed the Expert-button. | Boolean |

The view of the graphical user interface shall look like Figure 54.

6.1.3.3 Removing a Server Entry

In case that you want to remove server entries from your profile, you just have to select the „Server Entry Icon“ and press the second mouse button. A new menu will appear which allow you to remove a server entry from your profile.

The other alternative is to press directly on the „Remove Icon “ after you have selected the server entry (See Figure 54).

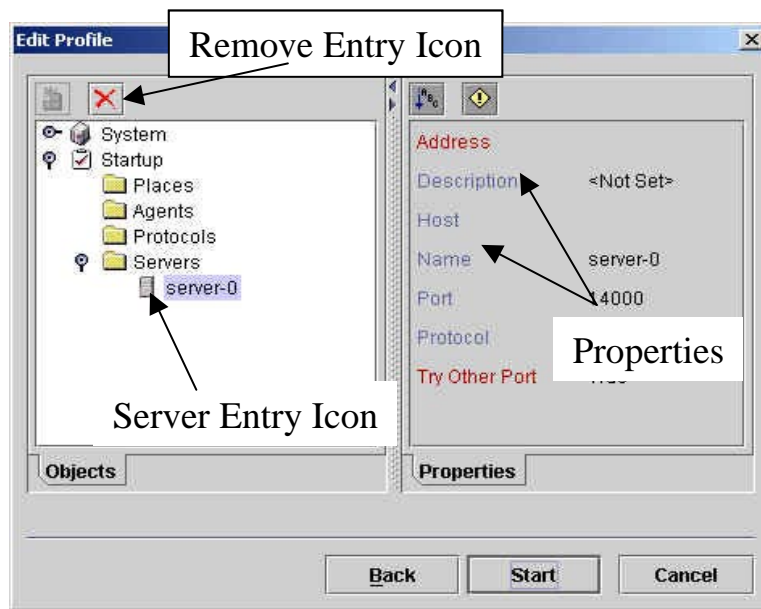


Figure 54: Server Entry Properties

6.1.4 Place Properties

Within the Profile, you can specify whether you need some places to be initially created right at the beginning of your Grasshopper session. The place entry is exactly made for this purpose.


The „Profile“-Editor enables you to add, modify and remove place entries.

A place entry will only be applicable to Agent System. Thus, will only appear for setting up your Agent System’s profile.



6.1.4.1 Adding Places

If want that the Grasshopper Assistant to create for you places initailly, you have to specified them in the profile.

A place specification can be added to the profile by selecting the „Place“-folder and either use the second mouse button or „Adding Place“-Icon  within the „Icon Area“.

Once you have done this, a dialog box will appear on your screen where you can enter properties related to a Place. There are two of them that you have to set. The first one, the name of the place, is mandatory. It will uniquely identify the place during a Grasshopper session. Thus make sure that you do not assign a name twice to places¹. The second one, the description, serves as a descrip-

tion of the place. You can put a text into this field that describes the purpose. This field is optional.

6.1.4.2 Modifying Place Properties

You can change your setting of the places any time you like. This is as easy as adding a new place to your profile.

First you have to select the place entry. After the selection, the related properties will appear at the right side of the graphical user interface, the „Object Property Area“.

The next, you just have to select the property value and by doing this, the respective field will change to a text-field, where you can make your desire modifications. The „Profile-Editor“ shall look like Figure 55.

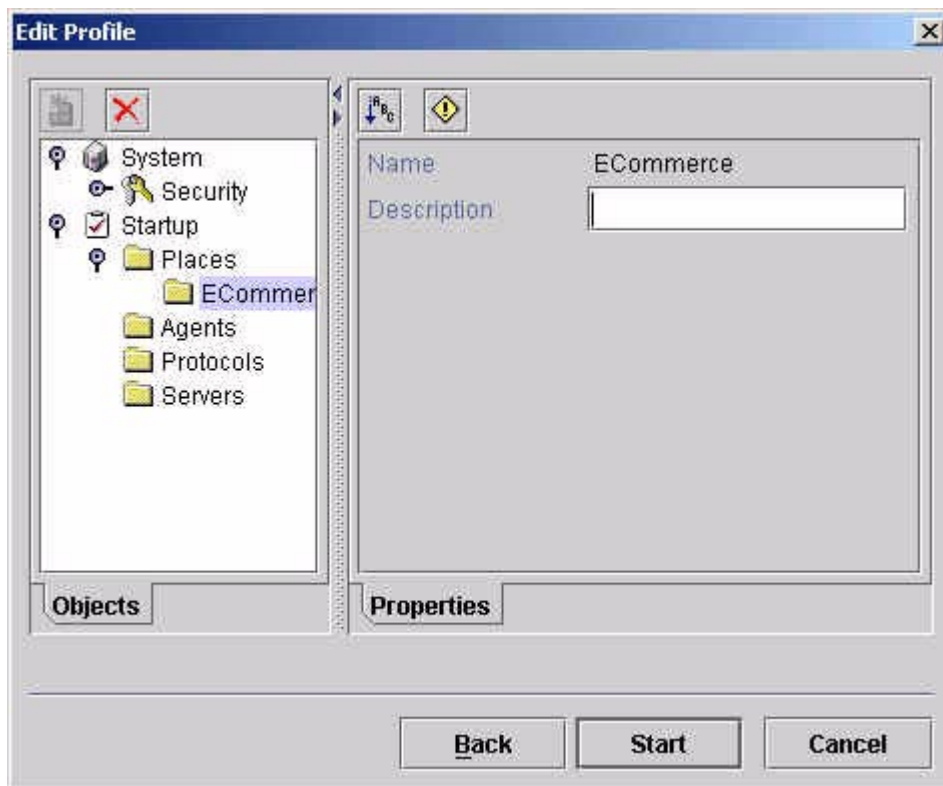


Figure 55: Modifying Place Properties

6.1.4.3 Removing Places

In order to remove a place entry from your profile, you have to select the entry

1. Right at the „Profile-Editor“ you can assign as many as places to a name. The Assistant will consider all of them as a single place, thus will only create a single place with that name.

and either uses the second mouse button to activate the remove menu or uses the „Remove -Icon“ in the „Icon-Area“.

6.1.5 Agent Properties


With Agent entry, you can define Agents that will be created during the start-up of Grasshopper.

You have to note that Agent entry will only appear when you define your Agent System profile. Thus, it is not applicable to a Region related profile.



6.1.5.1 Adding an Agent Entry

Creating an Agent entry follows the same pattern as creating the other entries.

Once you have selected the „Agent“-folder, a „Adding Agent“-Icon  will appear in the „Icon“-area. You can either use this to create an Agent entry or use the second mouse button to activate an „Adding Agent“ menu. Thereafter, a dialog box will appear on your screen.

This Agent dialog box contains a form, requesting properties for creating an Agents. Associated with an Agent entry, there are six properties you have to provide:

| Property Name | Purpose | Value Type |
|---------------|--|------------|
| Name | The <i>Name</i> property is only for labeling purpose. Thus, is used as an Identifier and therefore, must be unique. In this field, you can specify a name for the Agent. This name is used to identify the Agent within the profile. Please note that this name is not used to represent an Agent instance during runtime. You have to provide a name for an Agent. | String |

| Property Name | Purpose | Value Type |
|---------------|---|------------|
| Description | If you want to provide a some textual information related to the server, then you can put it right here, in this property. This field is optional. | String |
| Classname | The <i>Classname</i> is mandatory. In this field, you have to specify the fully-qualified Java-classname that represents the Agent. | String |
| Codebase | If you want that the code of the Agent shall be loaded from a particular code base, then you can specify in here by providing a URL. | String |
| Place | The Agent can be created initially at a particular place. The InformationDesk is the default case. If the creation of the Agent failed in the desired Place, the Agent will be created in the InformationDesk only if the creation was denied by a security policy. | String |
| Arguments | Any arguments that shall be passed to the Agent have to be specified within this field. You have to note that the Assistant will not check their validity, i.e., any errors will prompt to you not until the actual creation process. The arguments must be separated by a space. This property will only be shown if the expert-mode has been applied. | String |

The following Figure 56 shows the dialog box in which you can enter the value

for these property types.

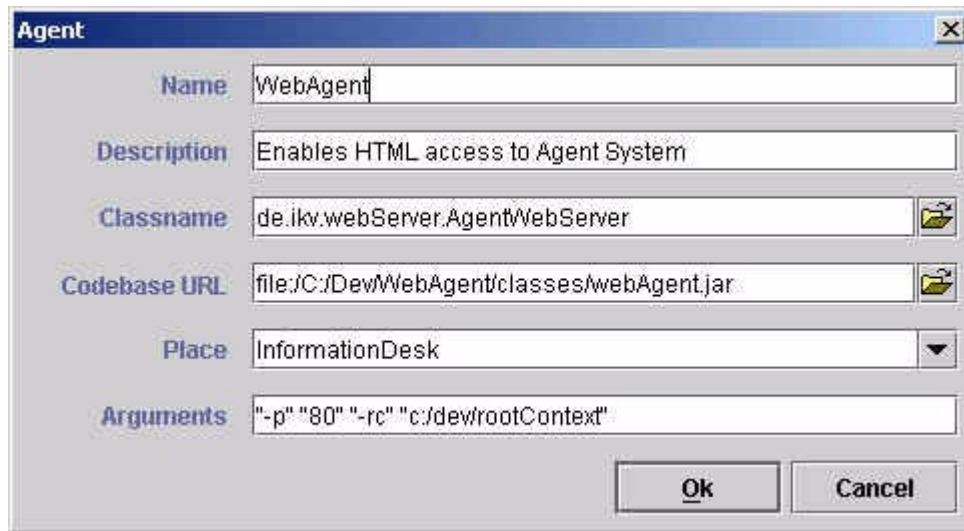


Figure 56: Adding an Agent Entry

6.1.5.2 Modifying an Agent Entry

If you want to modify your existent Agent entry you just have to select the „Agent“-entry beneath the „Agents“-folder. Once you have done this, the respective properties will be displayed in the „Object Property Area“.

There are six properties, which are the same as described in the previous section. In order to modify, you have to select the property value. When ever you move your mouse pointer over a value, a box will surround that value. By pressing the first mouse button you are able to start your modification.

6.1.5.3 Removing an Agent Entry

If you like to remove an Agent entry from your profile, you have to select the „Agent“-Entry beneath the „Agents“-folder. After the selection, you can either use the „Removing“-Icon at the top in the „Icon“-area or use the second mouse button to activate the „Remove“-menu.

As soon as the removal has taken place, the respective Agent icon will be removed from the graphical user interface. Also, the next time you start up an agency with this profile, it won't create this agent right at the start-up.

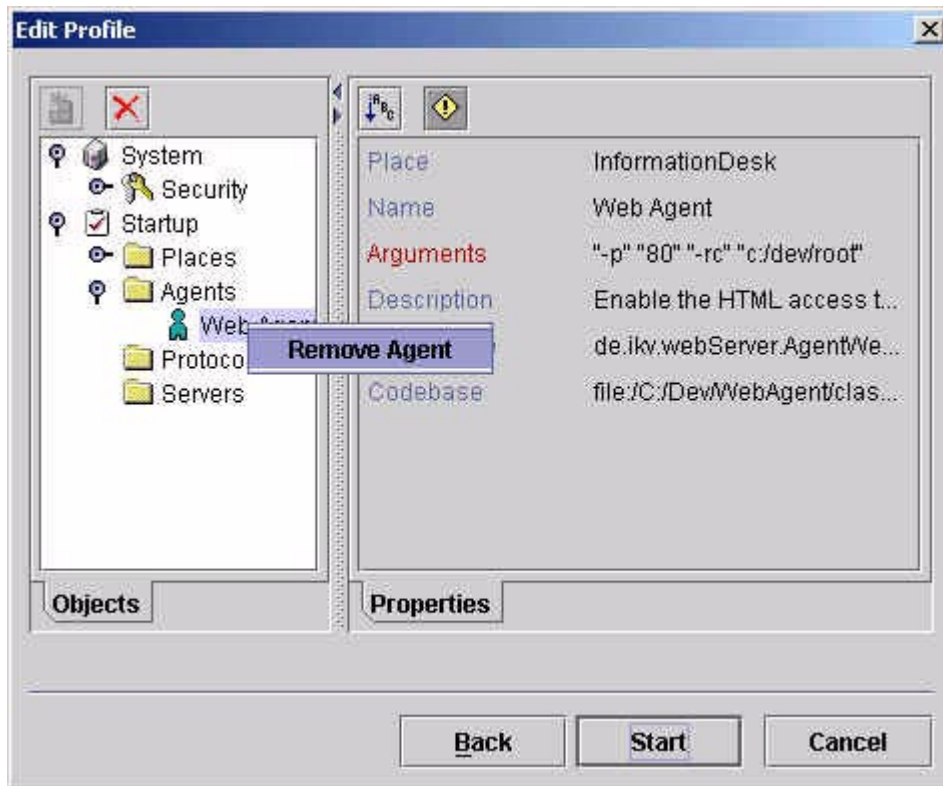


Figure 57: Removing an Agent Entry from Your Profile

6.1.6 Profile Structure

Grasshopper will store all the properties on your file system. The location of the profiles are specified during the first time invocation of the Assistant. As described in Chapter 3.1.3, you can provide a profile path for storing the profile values.

If no specification is done, the Assistant assumes that each user has a user home directory. This home directory is then used by the Assistant to store the user's profile.

The content of such a profile is depending on your setting. In general, a profile is closely following the pattern of the graphical user interface. For each property type: System, Places, Agents, Protocols, and Servers there will a respective string in the profile, such as, [System] or [Servers], a so called tags.

In addition, for each sub-type of the above property types, a string representation follows the pattern: [<property type><number>] is used. For example, for each Agent entry, you will find:

```
[agents]
agent0=<name of agent>
agent1=<name of agent>

[agent0]
...

[agent1]
...
```

All property values, as described within this chapter, that you have modified via the „profile“ graphical user interface can be found right after the respective tags.

You shall note that the profile will only include the property values that you have explicitly modified. The not touched one will not be stored within the profile, since they are default values.



The following Figure 58 just shows one example of a profile. It includes entries for places, servers, agents and system.

6.1.7 Modifying the Profile

The Grasshopper system will read in this profile on start-up. One way of changing the content is to use the Assistant as described above. A more direct and faster way of adjusting the profile is to directly load it into a text editor.

In the following, a set of examples will show you how it can be done. They will include adding new places and agents to the profile as well as changing and adding properties.

The starting point for the examples will be the default profile shown within Figure 59. The default profile is delivered with Grasshopper contribution.

First, load this profile into your favorite text editor and save it as, for example, myProfile.ini.

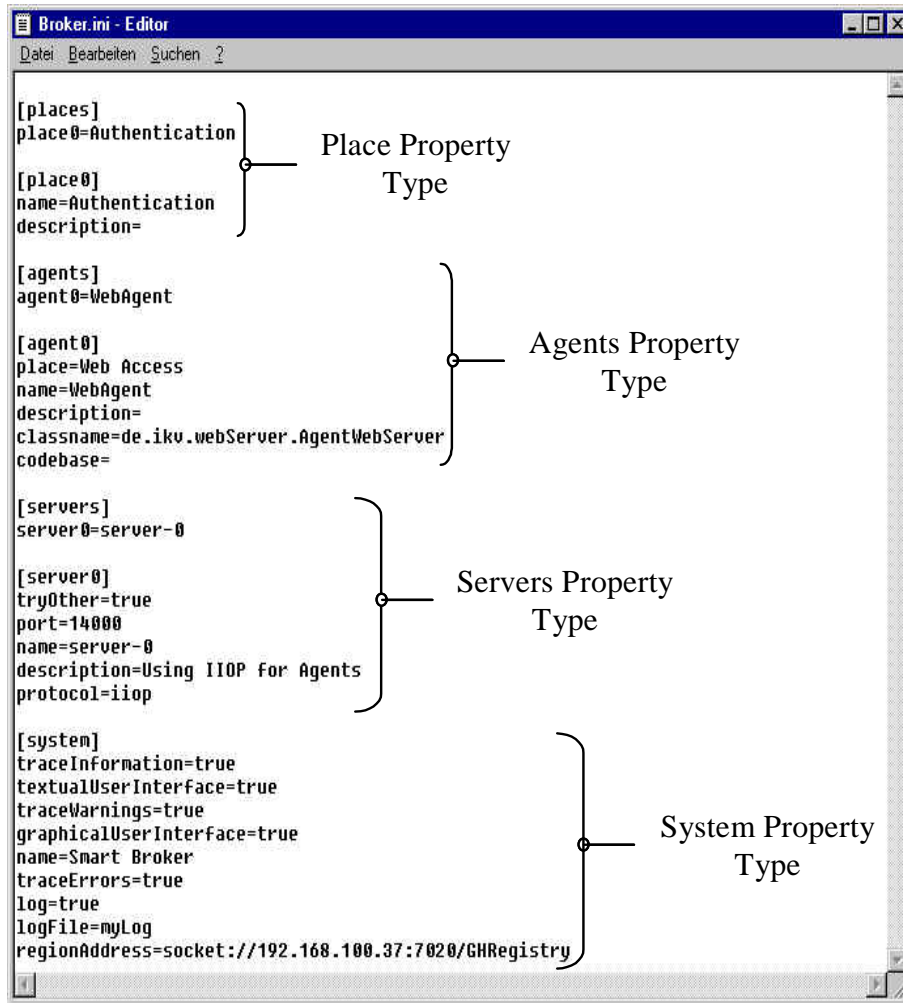


Figure 58: Structure of a Profile



Figure 59: The default Profile

6.1.7.1 Adding New Places

If your application need some places to be created in advance and automatically by the Assistant then you have to modify *myProfile.ini* in the following way:



1. Using a text editor and opens the file *myProfile.ini*
2. Adding beneath the last line of the *[system]*-block, the *[places]*-tag. This tells Grasshopper that some place specifications will follow. The *[places]* collects all place-entries and has the *place<number>=<Name of Place>* attribute. For each entries, the *<number>* has to be unique, since Grasshopper is using this for identification purposes.
3. Putting on the next line: *place0=myFirstPlace*
4. In order to specify the place entry *place0*, you have to add the *[place0]*-tag to the profile. This tag includes two attributes, one identifying the name of the place to be displayed within the agency's graphical user interface and the second provides a description of the place. The structure looks like as follow: *<name>=<Place name>* and *<description>=<text>*.
5. Putting the following value into the profile:
name=my first place
description=This is my first place to be created by the Assistant.
 The profile shall look like shown in Figure 60.

```

[system]
textualUserInterface=true
graphicalUserInterface=true
traceWarnings=true
traceErrors=true

[places]
place0=myFirstPlace

[place0]
name=my first place
description= This is my first place created by the Assistant.
  
```

Figure 60: myProfile

6. To create an agency with this profile use the following commands:
java de.ikv.grasshopper.Grasshopper a -pr myProfile. Thereafter, the fol-

Following Figure 61 agency shall be shown on your computer screen.

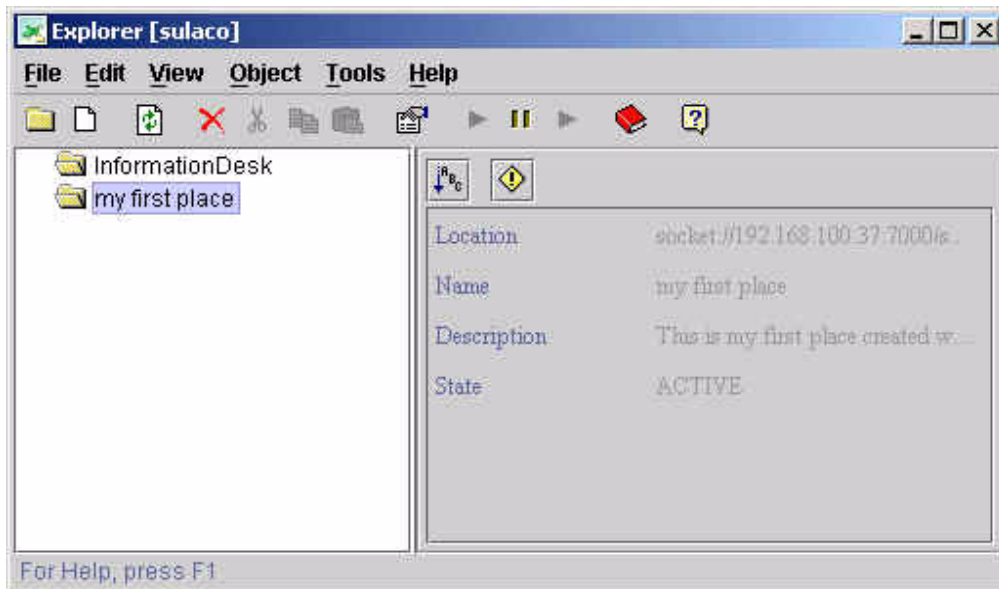


Figure 61: The Agency using myProfile

6.1.7.2 Adding New Agents

Corresponding to adding a new place is the adding of agent entries to the profile. For this purpose, some tags has to be used. The relevant tags are:

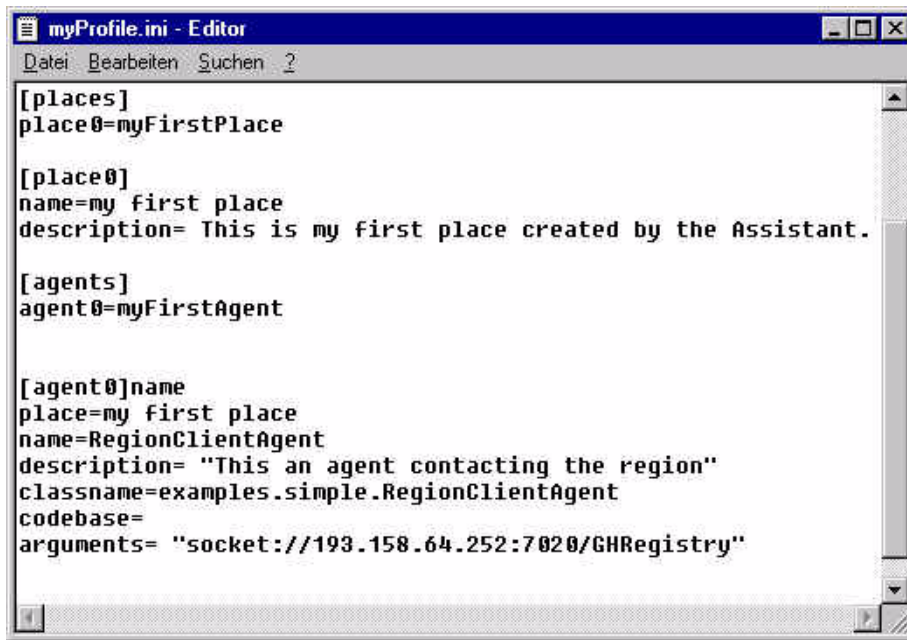
- *[agents]*: This group the agent entries. Same as *[places]*, the only attribute is *agent<number>=<name of agent>*. The number must be unique.
- *[agent<number>]*: Each entry stated in *[agents]* will have an corresponding tag that specified the attributes of an agent. The possible attributes are defined Chapter 6.1.5; these are *name*, *place*, *description*, *classname*, *codebase*, and *arguments*.



In order to add an agent entry to the profile, you have to add the following entries to your profile as shown in Figure 62.

This example takes one argument that identifies the Region Registry to be contacted. If you are using your own Region, the *arguments* attribute must be modified. Remember in order to create this example Agent, the CLASSPATH must be extended to point Grasshopper examples directory.

If you want to pass more arguments to your self-programmed Agents then you have to put them all on a line where each argument is enclosed by quotation marks and is separated by a space, e.g., „arg1“ „arg2“ „arg3“ ...§



```

[places]
place0=myFirstPlace

[place0]
name=my first place
description= This is my first place created by the Assistant.

[agents]
agent0=myFirstAgent

[agent0]name
place=my first place
name=RegionClientAgent
description= "This an agent contacting the region"
classname=examples.simple.RegionClientAgent
codebase=
arguments= "socket://193.158.64.252:7020/GHRRegistry"

```

Figure 62: myProfile including Agent entries

6.1.7.3 Adding Agency's Attributes

There are also some tags and attributes associated with an Agency. The complete set is described in Chapter 6.1.1. The following example shows only an excerpt, just giving you an idea what attributes for Agency you can set within the profile.

Properties related to an Agency is addressed with the *[system]*-tag. MyProfile.ini has such an entry which had already included four attributes. These covers interfaces and the types of information to be output.

The following example will show you how you can give an Agency a particular name and force it to write all outputs into a log file. It also shows you how you can put down the initial address of a Region. The Figure 63 shows all entries you have to add to your profile.

These attributes are involved in this example:

- *name=<name of the Agency>*: Put down the desire name of the Agency. For the example, this name is used: „My first Agency“.
- *log=<true/false>*: Set this attribute to true, in order to activate the logging feature.
- *logFile=<Name of the file>*: Write down the file name. The example uses D://users/dev/MyLogFile



- *regionAddress*=<Grasshopper Address>: Put down the address of the Region. The example shows the Region used by IKV.

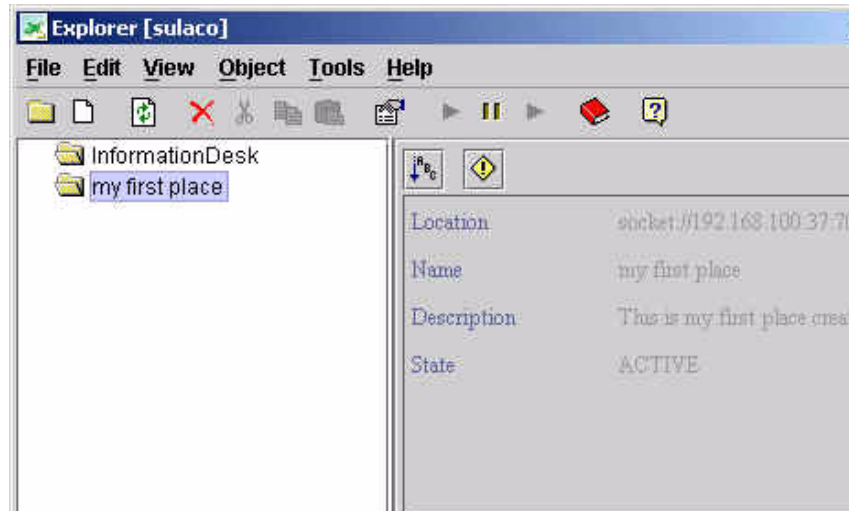


Figure 63: Extending Agency's Entry

6.2 Modifying Preferences

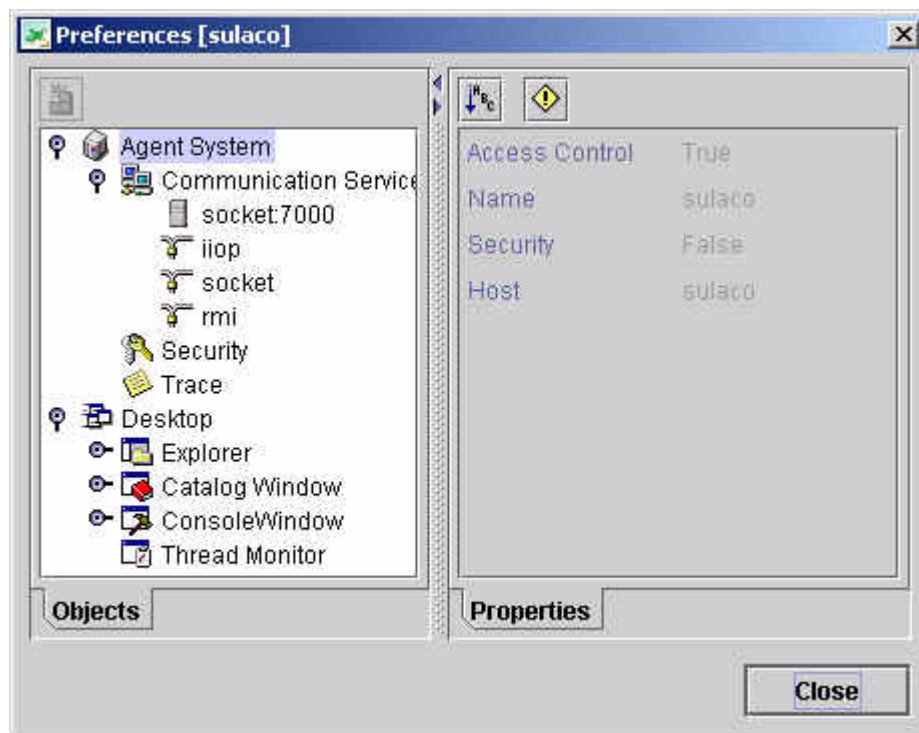


Figure 64: The Preference Window

A Grasshopper Program Switches

This appendix describes the usage of the command-line switches for starting up Grasshopper applications.

Grasshopper provides you a simple start-up program in form of a shell script, which enables you of launching different Grasshopper applications in a generic way. The term „Grasshopper application“ is a collective noun for the programs: Assistant, Agency and Region. Each time this collective noun is used, all the provided information applies to either programs.

In order to start a Grasshopper application you have to use the following command within a shell:

```
Grasshopper [Command] [Option]+ [Parameter]+ [--] [PassThru-Options]
```

You shall keep in your mind, that on Window 9x, you are only be able to pass 9 parameters via the command-line. This is due to the limitation of the Windows 9x.



The [Command] part is used to specify the type of application to be launched by the start-up program. With the next part, the [Options], you can declare setting values for the Grasshopper environment, in general. The [Parameter] part, you can indicate selective parameters to be used for configuring the to be launched application. To signify the end of Grasshopper related parameters, you shall use [--]. If you need to pass arguments to running environment, such as, CORBA or the to be launched applications needs arguments, you can specify them thereafter.

Switches Grammar: EBNF

The grammar specification, which uses EBNF as the notation, describing all the possible switches are as followed:

- (1) <grasshopper> ::= 'Grasshopper ' [<applications>]
- (2) <applications> ::= [<help>] | [{<assistant> | {<region> | <agency>}}]
- (3) <help> ::= '-h[elp] ';
- (4) <assistant> ::= ([<verbose>] [<path>]);
- (5) <region> ::= 'r[egion] ' ([<common-options>] [<region-options>]);
- (6) <agency> ::= 'a[gency] ' ([<common-options>] [<agency-options>]);
- (7) <common-options> ::= ([<name>] [<preferences>] [<io-options>])
- (8) <name> ::= '-n[ame] ' NAME
- (9) <preferences> ::= <profile> <path>
- (10) <profile> ::= '-pr[ofile] ' FILE

- (11)<path>::= 'pa[th] ' PATH
- (12)<io-options>::= [<verbose>] [<log>] [<tui>] [<gui>]
- (13)<verbose>::= '-v[erbose] '
- (14)<log>::= '-l[og] ' LOGFILE
- (15)<tui>::= '-t[ui] '
- (16)<gui>::= '-g[ui] '
- (17)<region-options>::= ''
- (18)<agency-options>::= [<region url>] [<persistence>] [<access control>] [<server>]+ [<place>]+ [<agent>]+
- (19)<region-URL>::= '-r[egion] ' URL
- (20)<persistence>::= '-pe[rersistence] '
- (21)<access control>::= '-ac[cess] '
- (22)<server>::= '-ser[ver] ' PROTOCOL [PORT]
- (23)<place>::= '-p[lace] ' PLACENAME [DESCRIPTION]
- (24)<agent>::= '-a[gent] ' CLASS[@PLACENAME] ([CODEBASE [ARGUMENTS]+])

Description of Each Command-line Option:

The following three tables provides a detailed description of each possible switches of each part:

| Command | Descriptions |
|---------|---|
| [] | If you invoke the start-up program with no argument, an Assistant will be started. This Assistant will guide you thru the start-up and customizing procedures of an Agency and a Region. Note, if you need to create or adjust your personalized profile, this way of starting up is recommended. More details about the usage of the Assistant, please refer to Chapter 3 in this documentation. |

Table 1: Command Flags

| Command | Descriptions |
|----------|--|
| r[egion] | <p>If you want to launch a Region, you have to use the r[egion] command parameter. In general, the Region will use default settings.</p> <p>If personalized settings are required, you have to use the Assistant (see above) or hand over a defined profile by specifying it with the -pr[ofile] <filename> and by setting the path to the profile with the -path parameter. The last input is only required if the profile is stored in a location which differs from the default location, i.e., the location where Grasshopper was initially installed.</p> |
| a[gency] | <p>Analog to the region command-flag, this will launch an Agency using the default settings.</p> <p>If other settings shall be used, you have to specific the profile with the -profile <filename> parameter or start the Agency via the Assistant.</p> |

Table 1: Command Flags

| Options | Descriptions |
|------------|--|
| -h[elp] | <p>By invoking the start-up program with this flag, a list of all possible switches along with a short description of it will be displayed to you on your screen.</p> <p>Note, any other hand-over command-line options along will be ignored. This holds for any Grasshopper applications to be launched.</p> |
| -v[erbose] | <p>By using this flag, some additional output will be shown. In general, only error messages will be displayed. If you activate the verbose mode also warning as well as further information will be provided.</p> |

Table 2: Option Flags

| Options | Descriptions |
|---|---|
| <p><code>-pr[ofile]</code> <code>FILENAME</code></p> | <p>This flag enables you to specify a profile to be used instead of the default one. Once specified the start-up program will take the preferences from the specified profile for setting up the newly created Agency or Region. The profile option will be ignored if the Assistant is used.</p> <p>The FILENAME is a must attribute and can therefore not be omitted. You can specify the FILENAME either absolutely or relatively. For example on a Windows System an absolute defined FILENAME may look like this: <code>d:/users/Agent/Shopping.ini</code>. A relatively defined name will be just <code>Shopping.ini</code>.</p> <p>For more details about the structure of a profile, please refer to Chapter 6.1.</p> |
| <p><code>-pa[th]</code> <code>PATH</code></p> | <p>You can provide a new path pointing to the profiles and preference files to be used for the start-up of an Assistant, Agency and Region.</p> <p>The PATH is a must attribute and has to be in the absolute form, pointing from root to the directory, where the profiles are stored.</p> |
| <p><code>-l[og]</code> <code>LOGFILE</code></p> | <p>This option will arrange Grasshopper to write out all warnings, errors, and additional information to the specified log-file named by LOGFILE.</p> <p>The LOGFILE name has to be provided and can be either stated absolutely or just named a file name. If only a file name is given, the default preference path is used as the storing location.</p> |

Table 2: Option Flags

| Parameters | Descriptions |
|----------------------------|---|
| <code>-n[ame] NAME</code> | In cases that you have to assign a name to an Agency or a Region explicitly, you must use this parameter. Note that this parameter is only applicable in combination with the command option <code>a[gency]</code> or <code>r[egion]</code> . No effect will take place with the Assistant; it will be simply ignored. The NAME is a string and can be in any combination of printable characters or numbers. |
| <code>-g[ui]</code> | Applying this flag in conjunction with the command <code>a[gency]</code> or <code>r[egion]</code> , the associated graphical user interface will be displayed on the screen. Just for your Information, you can also launch a graphical user interface via the <code>tui</code> -command. You just have to type in: <code>gui</code> within the command-line. |
| <code>-tu[i]</code> | If you apply the <code>-t[ui]</code> option, a textual user interface related either to an Agency or a Region will be started along, depending on which command option you were used. |
| <code>-r[egion] URL</code> | This option allows you to tell an Agency, the location address of an existing and running Region. The address is specified as a URL. The URL has the following representation: <code><protocol>://{<host>:<Port>}/<Region-Name></code> . For example, the following Strings are all valid URLs: <ul style="list-style-type: none"> • <code>socket://artemis:7020/GHRegistry</code> • <code>socket://122.122.122.122:7020/RegionRegistry</code> • <code>socket://localhost:13000</code> |

Table 3: Parameters Options

| Parameters | Descriptions |
|---------------------------------|---|
| -pe[rsistence] | <p>In order to enable an Agency to support persistency, this option shall be used. You won't see any graphical differences between an Agency running with or without the persistency service. This function will only affect Agents that are designed and implemented with persistency support, i.e., are a subclass of either PersistentMobileAgent or PersistentStationaryAgent.</p> <p>For more details, please consults the Programmer's Guide.</p> |
| -ac[cess] | <p>This option will turn on the security feature of an Agency. Once activated, the Agency will follow a security policy to supervise any actions carried out by any Agents.</p> <p>Note, you won't actually become aware that the security feature is activated except that of some security related exception are shown and a slight performance reduction of your agent-based applications.</p> |
| -ser[ver] PROTOCOL [PORT] | <p>You can initially define servers to be launched by the start-up program. The server option requires two arguments. One, the Protocol, is a must attribute while the second, the [Port], is optional.</p> <p>With the 'Protocol' you can select the type of the server to be started. In general, you can pick out a socket-server, iiop-server, rmi-server.</p> <p>Besides the selection of the protocol type, you can also specify the port to be used by the server.</p> <p>If you need multiple servers, you have to apply this option respectively. Please note that Grasshopper will start a socket-server on port 7000 on default. The same also holds for a Region. In this case the port 7020 is used.</p> |

Table 3: Parameters Options

| Parameters | Descriptions |
|--|---|
| <code>-p[place]</code> <code>PLACENAME</code> <code>[DESCRIPTION]</code> | <p>You can tell the start-up program to create initially a Place for you. The name of the Place is specified by <code>PLACENAME</code>. Any printable characters or numbers can be used.</p> <p>If more Places have to be created, a multiple application of this option is possible. Associated with a Place is also a description which can be applied by the <code>[Description]</code> field. Please note that you have to quote the description.</p> |

Table 3: Parameters Options

| Parameters | Descriptions |
|--|--|
| <pre>-a[agent] CLASS[@PLA- CENAME] ([CODEBASE [ARGUMENTS]+])</pre> | <p>Along the creation of the Agency, you can also create initially some Agents just by using the <code>-agent</code> option. To do this, you have to provide some arguments to this option.</p> <p>With the first one, the <code>CLASSNAME</code> which is mandatory, you specify the name of the class to be loaded. You have to provide the the fully qualify Java class name, i.e., including the package name.</p> <p>For example: <code>de.ikv.grasshopper.examples.HelloWorldAgent</code> is a valid input. In this context, please note that you do not have to provide the <code>' .class'</code> extension.</p> <p>If you want the Agent to be created within a particular place, you have to specify it with the option <code>[@PLA-CENAME]</code>. Please note that there is no space between <code>CLASS</code> and <code>@PLACENAME</code>. The given place must be existent, i.e., has been specified via the option <code>'-place'</code> (see above).</p> <p>For example: the definition <code>de.ikv.grasshopper.examples.HelloWorldAgent@Placel</code> is a valid specification.</p> <p>Besides the declaration of the class, you have to make sure that the <code>CLASS-PATH</code> environment parameter contains an entry, pointing to the byte-codes repectively. Another alternative to load the class of the specified agent is to provide a <code>CODEBASE</code>. A code base identify a location from where the class can be loaded.</p> <p>The codebase can for example look like this: <code>http://www.grasshopper.de/classes</code></p> <p>If your Agent requires some arguments, you can provide them after the codebase specification. One important thing you have to keep in mind, once you have to specify some arguments, the codebase has to be declared as a URL or be an empty string.</p> |

Table 3: Parameters Options

B Grasshopper TUI Commands

Besides the graphical user interface, you can also use the textual user interface to access Grasshopper functionality. Such an interface enables you to invoke commands in an environment with no graphical supports.

Both Agency and Region have such a textual interface. In the following, the commands recognized by both interfaces will be explained.

The first section identifies common commands applicable to both, while the second and third part only to specific ones for Agent-System and Region.

B.1 Common TUI-Commands

There are a set of commands applicable to both, the Agent-System and Region. The list of the following commands are defined:

- Close, Annex B.1.1
- Exit, Annex B.1.2
- Gc, Annex B.1.3
- Gui, Annex B.1.4
- Help and ?, Annex B.1.5
- Info, Annex B.1.6
- List, Annex B.1.7
- Remove, Annex B.1.8
- Status, Annex B.1.9
- Thread, Annex B.1.10.
- Histroy and related, Annex B.1.11
- Trace, Annex B.1.12

B.1.1 Close

The „Close“-command enables you to close only the textual user interface without terminating the process of an Agent-System or a Region, i.e., you only close your the access point to it.



You have to be aware of that once you had closed a textual user interface, you might have lost any access points to either an Agent-System or Region.

For example, if you have started an Agent-System only with the textual user interface, your only entry point to the Agent-System functions is provided by the this. Just by closing it, you will lose any control possibility, permanently.

Thereafter, you can either let the Agent-System alive or use system specific command to terminate it.

B.1.2 Exit

The „Exit“-command will end the process of an Agent-System or a Region. Any used resource will be freed.

If called on an Agent-System, the „Exit“-command will also deregister all components (Places, Agents, and its own Agent-System entry) from the Region.

B.1.3 Gc

Once invoked the „Gc“-command, the garbage collector of the underneath virtual machine will be activated, allowing it to clean up and free not used resources.

This command has no parameters and no output to be printed on the your screen.

B.1.4 Gui

This will activate the graphical user interface of the component. The „Gui“-command has no further parameter.

You can find more usage information related to an Agent-System Window in Chapter 4.3. Within the same chapter at Section 4.4 you will find information for a Region Registry.

B.1.5 Help or ?

The „Help“ or „?“-command prints some help messages. Without providing any options, this will give you an overview of all applicable TUI-commands.

If you need usage instructions for a particular command, you can type in as an option to the „Help“-command, for example: help list.

The syntax of this command is:

```
{help|?} [<command-name>].
```

B.1.6 Info

The „List“-command provides you some overall information. If you need detail information of a particular component then you have to use the „Info“-command. By specifying the item number, the associated detail information will be displayed.

The syntax of this command is:

```
info [#item number]*
```

If you want multiple detail information, you can do that by providing multiple item numbers.

B.1.6.1 Output of Info

The „Info“-command can be applied to all components. Since each component has its own set of attributes, the view of the output vary depending on the selected component.

The following described attributes can also be used for the „List“-command to construct a filter for selecting or searching only particular information. The attributes are the same as those Keys described in Annex B.1.7.



Attributes of Place

Once you apply „Info“ on an entry of a Place, the following attributes as depicted within Figure 1 will be displayed:

- Name: Name the place name.
- Description: Provide textual description of the place.
- State: Tell the state of the place either active or in-active
- Location: Specify the location address of the place (see Annex B.1.7 for the syntax description)

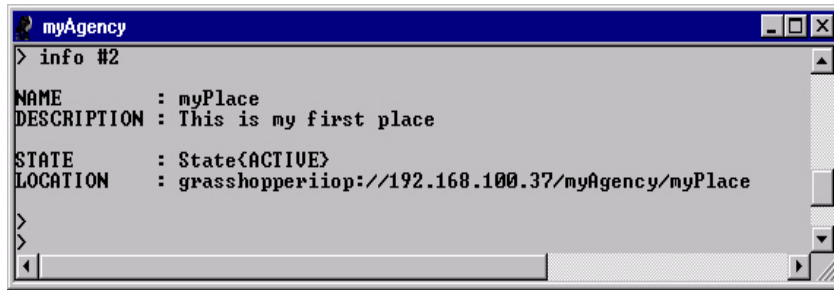
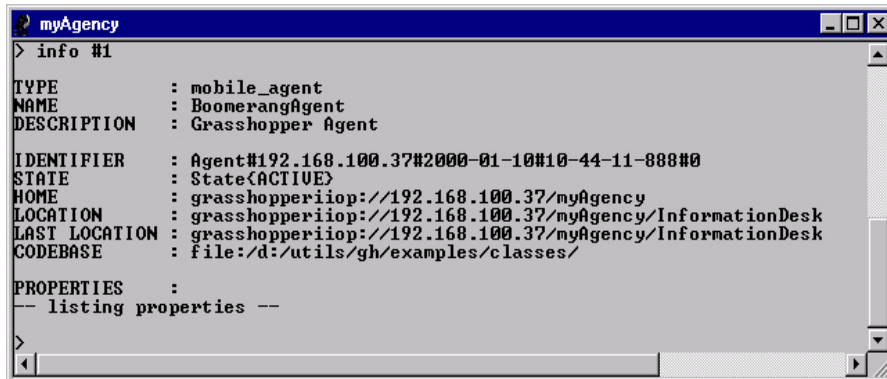


Figure 1: Detail Information Related to a Place

Attribute of Agent

Applying „Info“ to an Agent, the following information as depicted in will be provided to you on the screen:

- Type: Identify the type of the agent, either mobile or stationary
- Name: Name the agent name
- Description: Provide textual description of the agent
- Identifier: Contain the Agent Identifier
- State: Tell the state of the Agent, either active or inactive
- Home: Identify the location, where the agent was created (see Annex B.1.7 for the syntax description for a location)
- Location: Specify the current location of the agent (see Annex B.1.7 for the syntax description)
- LastLocation: Name the previous location from where the agent come from
- Codebase: Declare the location from where the code of the agent can be loaded (see Annex B.1.7 or the Programmer's Guide for more detail)
- Properties: List the properties of an Agent



```

myAgency
> info #1
TYPE           : mobile_agent
NAME           : BoomerangAgent
DESCRIPTION    : Grasshopper Agent

IDENTIFIER     : Agent#192.168.100.37#2000-01-10#10-44-11-888#0
STATE         : State{ACTIVE}
HOME          : grasshopperiop://192.168.100.37/myAgency
LOCATION       : grasshopperiop://192.168.100.37/myAgency/InformationDesk
LAST LOCATION : grasshopperiop://192.168.100.37/myAgency/InformationDesk
CODEBASE      : file:/d:/utils/gh/examples/classes/

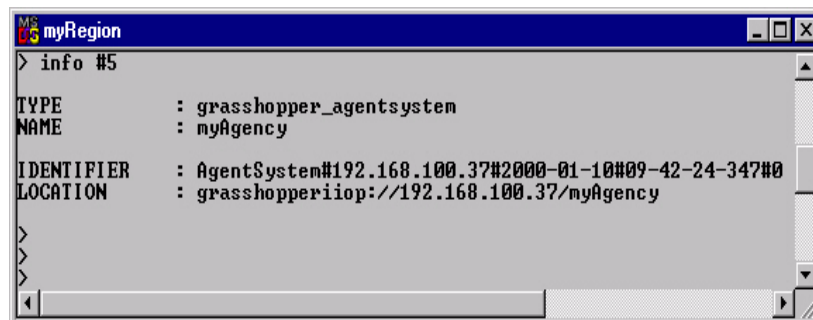
PROPERTIES    :
-- listing properties --
>

```

Figure 2: Detail Information Related to an Agent

Attributes of Agent-System

Applying this command to an Agent-System, the following output as depicted in Figure 3 shall be appeared on your screen:



```

myRegion
> info #5
TYPE           : grasshopper_agentsystem
NAME           : myAgency

IDENTIFIER     : AgentSystem#192.168.100.37#2000-01-10#09-42-24-347#0
LOCATION       : grasshopperiop://192.168.100.37/myAgency

>
>
>

```

Figure 3: Detail Information Related to an Agent-System

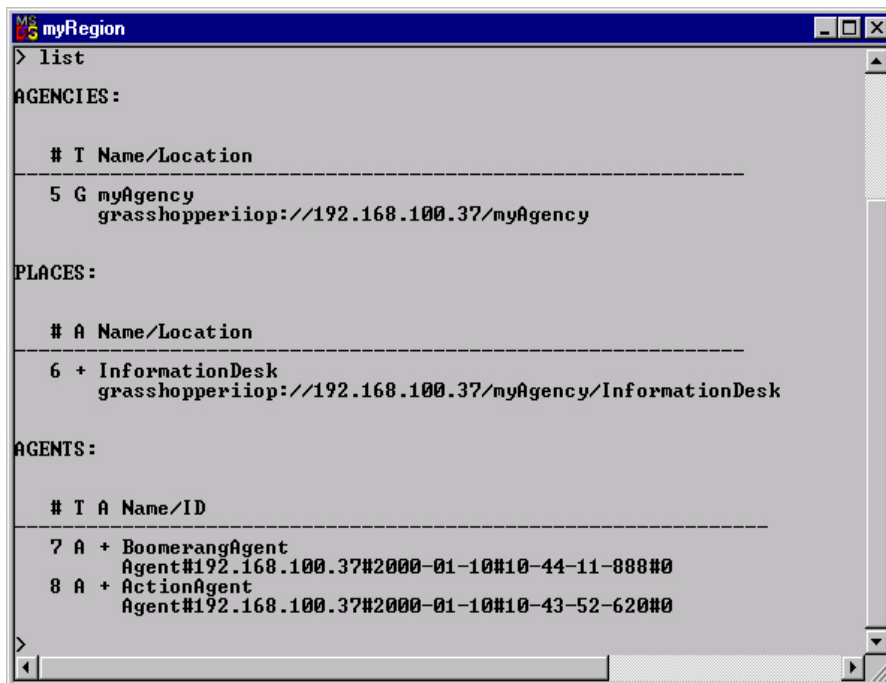
There are four attributes related to an Agent-System entry:

- Type: Identity the type of the Agent-System
- Name: Tell the name of the Agent-System
- Identifier: Specify the unique identify of an Agent-System
- Location: Declare the address of the Agent-System.

B.1.7 List

The „List“-command enables you to view the existing agents and places at a current Agency, if you apply this to an Agent-System. To get an overview of all existing components within a region, you have to use the „List“-command provided by the Region.

The following Figure 4 shows an exemplary output of this command, applied from a Region's textual user interface, on your screen.



```

myRegion
> list
AGENCIES :

# T Name/Location
-----
5 G myAgency
  grasshopperiiop://192.168.100.37/myAgency

PLACES :

# A Name/Location
-----
6 + InformationDesk
  grasshopperiiop://192.168.100.37/myAgency/InformationDesk

AGENTS :

# T A Name/ID
-----
7 A + BoomerangAgent
  Agent#192.168.100.37#2000-01-10#10-44-11-888#0
8 A + ActionAgent
  Agent#192.168.100.37#2000-01-10#10-43-52-620#0

```

Figure 4: List Command

You can use options with this command. There are two usage syntaxes, depending whether you are calling this via the Region's or Agent-System's textual user interface:

Syntax for Region

```
List [agencies : places : agents] [„filter“]
```

The command provides three selectors from where you can choose from. The 'agencies' one will only display entries related with Agent-Systems. The 'place' or 'agents' does the same, but only list respective information.

A more advance information selection mechanism can be used. You can build your own search filter (see below for more information).

Syntax for Agent-System

List [places : agents] [„filter“]

In order to list only the created places at the current agency, you must apply the „place“-option. By using the „agent“-option, only agent related information is listed. These two options will cover probably most of your use cases.

B.1.7.1 The Filter Construct

But however, if you need a more detail listing mechanism, you can use a so-called „filter“ to define the exactly type of information your are looking for. At the beginning, the „filter“-construct look very complex, but it isn't.

The grammar, specified in EBNF, of the filter is specified in terms of a `String` whose content must match to the following syntax:

```

Filter      = not-filter
not-filter  = ('!')? or-filter*
or-filter   = and-filter ( '|' and-filter )*
and-filter  = item ( '&' item )*
item        =  key comperator value
              | '(' not-filter ')'
comperator  =  '='          # equals
              | '^'        # starts with
              | '$'        # ends with
              | '~'        # contains
key         = java.lang.String
value       = java.lang.String

```

The minimal construct is „item“. By resolving item into its components, one will get the three components: a key, a comperator and a value:

```
item = key comperator value
```

These components are terminal symbols and can be therefore map onto character values. The key value represents the type, the comperator is a binary operator, and the value contains the information to be matched.

Keys

The following table lists all available keys and the components to which they can be applied to.

Note that some keys cannot be applied to all kinds of Grasshopper components (i.e., agencies, places, and agents). For instance, the `CODEBASE` key can only be applied to agents, and the `LASTLOCATION` key is only valid for mobile agents. If, for example, a list method is invoked with a filter that contains the `LASTLOCATION` key, all Grasshopper components that cannot be applied to this key are automatically excluded from the search. Thus, a method call that



uses the LASTLOCATION key within its filter can only return a list of mobile agents.

| Filter key | Description |
|-------------|---|
| CODEBASE | <p>This key identifies the code base of an agent. Such a code base tells the system from where to load the code of the agent.</p> <p>In general, a code bath is structured as followed:</p> <pre data-bbox="646 734 1299 801"><protocol>:[<path> or {<path>/<file>}].</pre> <p>The corresponding value must match to the code base syntax defined in Section 5.3 of the Programmer's Guide.</p> |
| DESCRIPTION | <p>By applying this, the textual description field of the searched component is used for the matching criteria.</p> |
| HOME | <p>If you want to find components from a particular home, this key shall be used. Note that this key is equal to LOCATION for stationary agents.</p> <p>The home key contains the host, the name of the Agency and -optionally- the place name only if the Agent was created within a place different from the „Information-Desk“.</p> |

Table 4: Filter Keys

| Filter key | Description |
|-------------------|--|
| LASTLOCATION | <p>If you want to list components that have been migrated from a particular location to the current Agency, this key has to be defined within the filter.</p> <p>The syntax of the corresponding key value must match to a textual representation as defined in Section 5.4. of the Programmer's Guide</p> |
| LOCATION | <p>The LOCATION key specifies the current location of the searched component. By using this key, you are able to select components based on their current location.</p> <p>The syntax of a value for this key must match to a textual representation as defined in Section 5.4. of Programmer's Guide</p> |
| NAME | <p>This represents the name of the searched component.</p> |
| SERVICEID | <p>Each searchable component is assigned to an identifier. This identifier is unique and can be used to uniquely addressing a component.</p> <p>The syntax of for this is defined in Section 5.1. of the Programmer's Guide.</p> |

Table 4: Filter Keys

| Filter key | Description |
|------------|---|
| STATE | <p>Current state of the searched component.</p> <p>A value of this key corresponds to the return value of the method <code>getState()</code> of the class <code>AgentInfo</code> or <code>PlaceInfo</code>.</p> |
| TYPE | <p>Type of the searched component.</p> <p>A value of this key corresponds to the return value of the method <code>getType()</code> of the class <code>AgentInfo</code> or <code>AgentSystemInfo</code>.</p> |

Table 4: Filter Keys

Comperator

You can use the following comperators to define your search constraint:

| Comperator | Description |
|------------|--|
| '=' | <p>If you are looking for a specific information that which shall match to a particular value, use the equal comperator. For example,</p> <pre>list „name=ActionAgent“</pre> <p>will list only agents named „ActionAgent“.</p> |
| '^' | <p>With the start-with comperator, you can list the information that starts with the value you have specified along. For example,</p> <pre>list „name^Act“</pre> <p>will list all entries that have „Act“ as the beginning sequence of characters.</p> |

Table 5: Comperators

| Comperator | Description |
|------------|--|
| '\$' | <p>Analog to start-with, you can also use the end-with comperator, if you want to find information that ends with a particular sequence of characters. For example:</p> <pre>list „name\$gent</pre> <p>will list all entries that contains in its name the ending sequence „gent“.</p> |
| '~' | <p>For searching information that contains a particular sequence of characters, the contain-with comperator shall be used. For example,</p> <pre>list „name~168.17“</pre> <p>All entries containing the specified values will be displayed.</p> |

Table 5: Comperators

B.1.7.2 Output of List

The output of the „List“-command is organized in a table which is shown in Figure 4. Within this figure, there are some used symboles, which needs to be explained:

- The '#' labels the column that contains the identifier of the row. The given value, or also called „Item Number“, is to be used to identify the entry. For example, you have to use this „item“-number for the „Remove“ or „Info“-command.
- The 'A' symbole represents the column intended for holding state information. A '+' means in this context that the named object is currently active. Respectively, a '-' reflect the resume state.
- The 'T' symbole will only appear for Agent or Agent-System related entries. For agents, if a 'S' appears in this column, it means that the addressed Agent is stationary and if 'M' is displayed then it is a mobile one. Related to an Agent-System, a 'G' means that your are dealing with a Grasshopper Agent-System.
- The 'Name' represents the textual identifier of a component (Place or Agent).

- The 'Location' specifies the address of the place. For example, this has to be provided to the „Move“-command, telling the system where to move an Agent.

A location specification has the following form: <protocol>://<host>/<agencyName>/<PlaceName>.

- The 'Agent Identifier' is unique and has the following syntax: <type>#<home>#<Date>#<Time>#<GenerationId>, where the <home> represents the host address that an Agent was created and <GenerationId> tells you whether the Agent is a master or a copy. The <GenerationId> follows such a number scheme: <master>.<CopyFrom-Master>.<CopyfromthefirstCopy>.<CopyFromTheSecondCopy>'...'. For example, the third generation copy from the master will have 0.1.1.1 as its generation identifier. The second copy of the first copy will have 0.1.2.

The following will illustrate these described symbols.

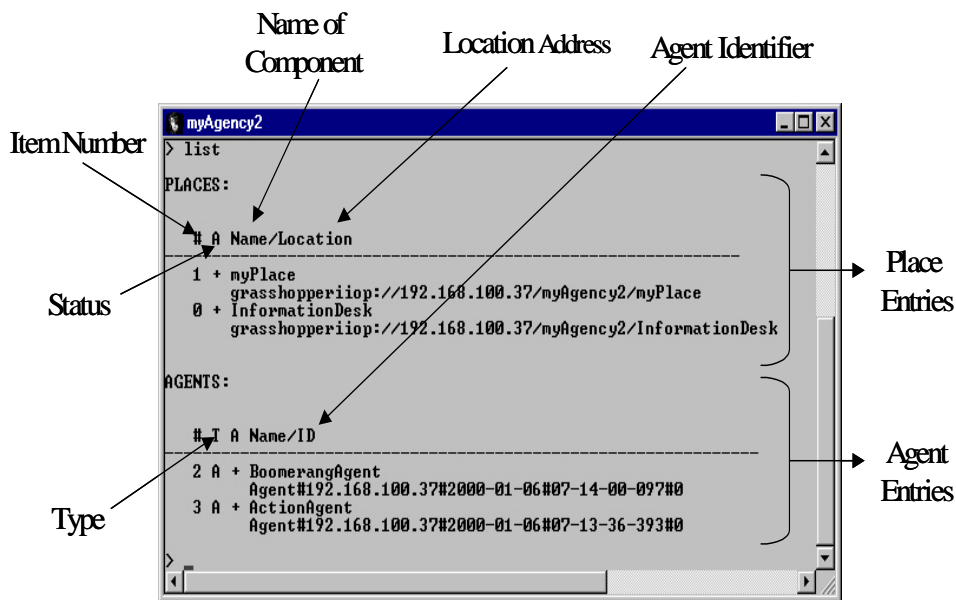


Figure 5: List Output Description

Places Entries

The output syntax for a place is comprised of three entries:

- The first one identifies the item number. This number is used by the TUI to identify an entry and must be provided by to other commands.
- The second entry tells you about the state of the place. A place can be

active, represented by '+', or inactive, represented by '-'.

- The third entry specifies the name and the location specification of the place.

For example, a valid entry for a place shall look like this:

```
3 + myPlace
  grasshopperiop://194.100.156.33/myAgency/myPlace
```

Agent Entries

The output syntax for agents is made up of four entries:

- The first one represents the item number. This must be used by other commands, such as „Remove“, to identify the selected entry.
- The second entry identifies the type of the agent. Within Grasshopper, two types are supported: one for stationary agent, which is represented by 'S', and two for mobile agents, which has the 'A'-symbol.
- The third one reflects the state of the agent. An Agent is either active ('+') or inactive ('-').
- The last entry specifies the Name and the AgentIdentifier.

Agent-System Entries

The output entry syntax for an Agent-System is quite similar to the agent one. It contains four entries:

- The first one specifies the item number of the entry.
- The second one tells you the type of the Agent-System. Mostly, the symbol 'G' will appear. This stands for a Grasshopper Agent-System.
- The third part identifies the name of the Agent-System.
- The fourth declares the address of the Agent-System.

B.1.7.3 Example

In order to be a little bit more familiar with the „filter“, you can see some examples in this section:

- Listing agents which are created at a particular Agency:

```
list agent home~<IP-Address>
```

- Listing all active places within an Agency:

```
list place state=active
```

- Listing all mobile agents:



```
list agent type~mobile
```

- Listing agents that are from a specific location:

```
list agent location~<IP-Address>
```

B.1.8 Remove

The „Remove“-command enables you to delete Agents, Places, and Agent-System. The syntax for this command is:

```
remove [#item]+
```

The only parameter for this command is the item number identifying the entry to be removed. You can retrieve the item number of an entry via the list-command, which is displayed in the column marked with the '#'-symbols.

You can combine multiple remove request into one just by providing all item numbers of those entries, lined into a sequence of item numbers.

```
remove #3 #2 #6
```



By applying this command within the Region's textual user interface, only the entries are deleted from the internal registry, i.e., Agents, Places, and Agent-System are still existing; they are just not be registered anymore.

There is a very important difference between the „Remove“ applied within the Agent-System's textual user interface and the one provided from the Region. Once used on an Agent-System for deleting either places or agents, they are actually deleted from the system. Also, this command deregisters them from the Region.

You shall note that the item number '#0' is reserved for the default place, the information desk, which cannot be removed.

B.1.9 Status

The „Status“-command displays information related to the component (Agency and Region). This command takes no parameter.

Depending on whether you call this command within the Agency or Region, a different list of attributes will be shown.

Attributes for Agent-System:

The „Status“ will print the following attributes on the screen:

- Identifier: This identifies uniquely an Agent-System. The structuring

pattern follows the same rule as for `AgentIdentifier`, which was described in Annex B.1.7.

- **Alive:** This tells you about the state the Agent-System. It can represent two value: either true or false.
- **Servers:** This entry declares the number and types of active communication servers at the current Agent-System. The location address of each server is also provided.

An Agent-System can held a list of communication servers. A default socket server will be started by the communication service at port 7000 during the start-up.

Grasshopper initially supports three types of communication server: socket, iiop, and rmi. You can extent Grasshopper with your own server, in this case please refer to the respective chapter within the Programmer's Guide (???).

- **Protocols:** This entry names the supported protocols. In a default case, iiop, rmi, and socket are supported.
- **Memory:** The memory attribute informs you of the current memory usage.

Attributes for Region:

- **Servers:** The same as described for an Agent-System, this entry identifies the active servers at the current Region.
- **Protocols:** This declares the supported protocol at the current Region.
- **Memory:** This will inform you about the current memory usage of the region.

B.1.10 Thread

The „Thread“-command will print out all threads of a component, which can be either an Agent-System or a Region.

The following shows the outputs of this command. The information provided in each row follows the pattern:

`<Type>: <NAME>;' 'Priority:<Value> ['Daemon'] <State>`, where

- `<Type>` tells you whether the information is about a „Thread“ or a „Thread Group“.
- `<Name>` identifies the name of the instance of the `<Type>`.
- `'Priority:<Value>` declares the priority which the instance of `<Type>` is

running.

- ['Daemon'] is optional and is only provided if the thread is a daemon thread.
- <State> contains state of information of the instance.

<Type>:<Name>;'Priority:'<Value> [Daemon] <State>

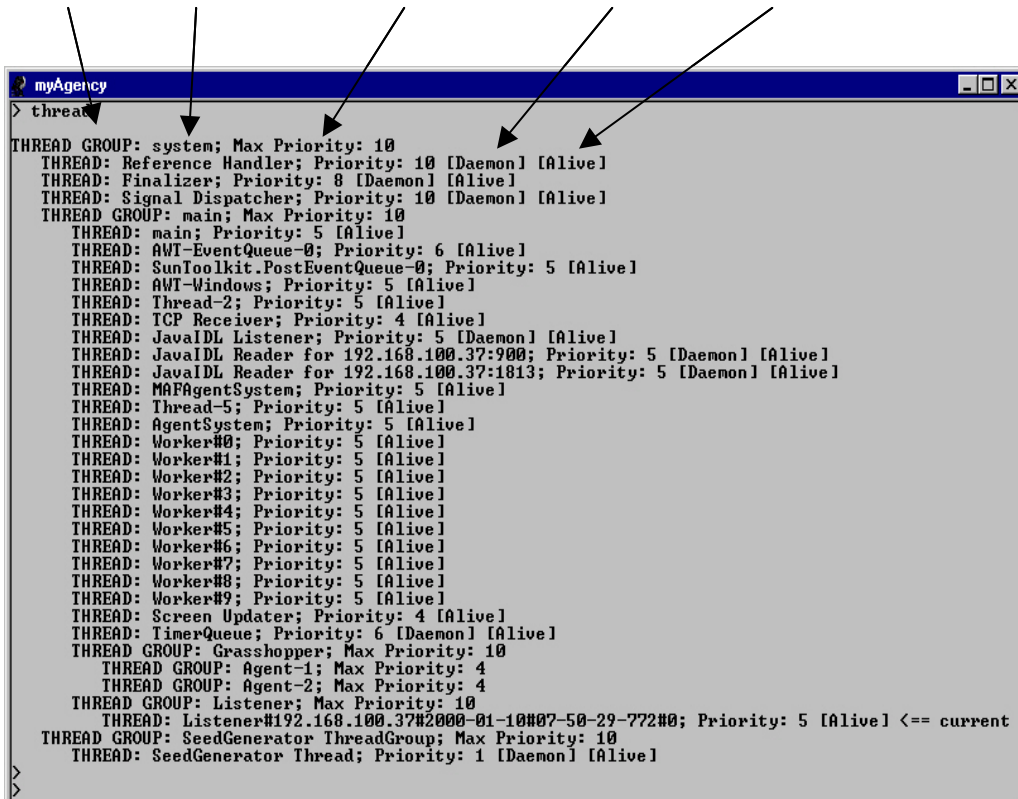
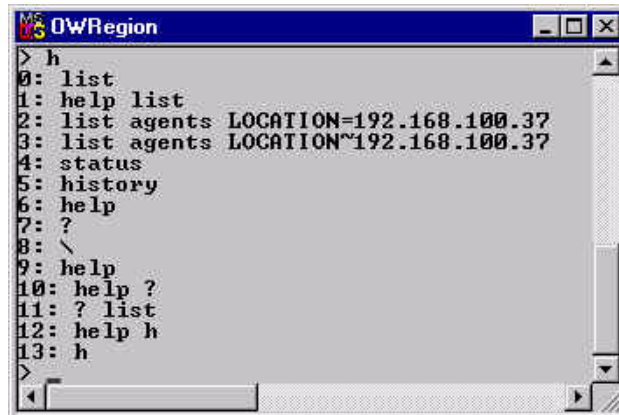


Figure 6: Outputs of the „Thread“-Command

B.1.11 History and Related

The History command provides an efficient mean to reuse your recent called commands within the Tui. Once called, a list of called commands will be displayed. Each command is assigned to a number, which can be used in conjunction with the command „!““. For example, and assuming the history provides the following outputs as shown in Figure 7:



```

OWRegion
> h
0: list
1: help list
2: list agents LOCATION=192.168.100.37
3: list agents LOCATION~192.168.100.37
4: status
5: history
6: help
7: ?
8: \
9: help
10: help ?
11: ? list
12: help h
13: h
>

```

Figure 7: History Command

With the „!“-command you can type in for example „!3“ and all agents which are from location with the IP-Address: 192.168.100.37 will be shown to you.

If you just want to repeat your last called command, you have to use „!!“.

For summary, the called syntax for is as followed:

- history
- !<number>
- !!

B.1.11.1 Modifying the History Files

Grasshopper will store a history file for each created Agent System. The differentiation will be made in accordance to the given name of the Agent System, i.e., each Agent System with a particular will have an associated history file, storing the last executed commands. The number of stored commands is defaultly set to 100.

Remember, the history file will only be created by the textual interface and not by the graphical user interface.

For Windows NT, the history file can be normally found under `c:/winnt/profiles/<username>/grasshopper`. If you are using Windows 2000, this file is located under `c:/Documents and Settings/<username>/grasshopper`. There is one exception for the location of the file, it is reported that if you are logged in as root and for some circumstances, this file is either not created or save to another location.





For Unix system, the preference file can be found under
 /user/local/<username>

B.1.12 Trace

The *Trace* command enables you to filter the type of information, such as, errors, warning, or information to be displayed either within the console or the graphical user interface.

The Syntax of this command is specified as followed:

```
trace [on|off] [error|warning|info]
```

By providing just the trace command, the current trace setting will be shown to you. No other arguments must be used.

The meaning of the arguments of this is described within the Table 6.

| Arguments | Description |
|-----------|--|
| on | This arguments will turn on the tracing feature of Grasshopper. In connection with no further argument (see below), it will enables all tracing information. In order to filter a particular type, you have to combine this with either error, warning, or info. |
| off | This will switch off the tracing feature in general, if it is applied with no additional argument. To selective removing a particular type, you have to provide it with this command. |
| error | This argument identifies the error type of information. |
| warning | This argument identifies the warning information. |
| info | Any other information which are not assigned to the two above will be associate with this type. |

Table 6: Arguments of the Trace Command

The type of an information will be determined by the program designer. On the code level, Grasshopper provides an application programming interface, enabling the categorization of the information. For this issue, please have a look at the Programmer's Guide or the API specification.

B.2 Agent-System Specific Commands

Additionally to the common commands as already described within the previous sections, an Agent-System provides the following extra commands:

- Create, Annex B.2.1
- Copy, Annex B.2.2
- Invoke, Annex B.2.3
- Move, Annex B.2.4
- Resume, Annex B.2.5
- Suspend, Annex B.2.6
- Reregister, Annex B.2.7

B.2.1 Create

The create command enables you either to instantiate a new Agent or to create a new Place in the Agency. The command has the following grammar:

```
c[reate]
{'agent' <CLASSNAME>['@'<Place>] {[CODEBASE] [ARGUMENTS]}} |
{'place' <PLACENAME> [<Descriptions>]} |
{'listener' <CLASSNAME> {[CODEBASE] [ARGUMENTS]}}
```

Creating an Agent

For creating an Agent, you have to provide the fully-qualified Java-class name of the Agent. Note that the extension '*.class' must be omitted. Optionally, a [CODEBASE] can be specified but only if the CLASSPATH environment parameter is not pointing to the agent's classes. If you wish the Agent to be created at a particular place then you have to provide the [PLACENAME]. Any required arguments for creating the agents can be passed to via [ARGUMENTS].

The first example will create an Agent. The CLASSPATH contains a pointer to the respective classes.



create agent examples.ActionAgent.

If you do not want modify your CLASSPATH, the following call will just do the same:

```
c agent examples.ActionAgent file:/d:/grasshopper2/examples/classes,  
if the agent classes are stored in d:\grasshopper2\examples\classes. Note that  
'c' is the short form of 'create'.
```

Creating a Place

You can also create a place within an Agency using the same 'create'-command. For this purpose, you have provide 'place' to the 'create'-command along with the name of the place. Optionally, you can a description of the place. This has no influence to the function of the place, it is just for information purposes.



For example, if you want to create a place with the name „MyFirstPlace“ then the following call will do the job.

```
create place MyFirstPlace „My first test place.
```

Creating a Listener

Besides a Place and an Agent, you can also create your own listener, e.g., to catch agent specific events and to customize their graphical representation. You can also automate certain reaction to those caught events.

For the creation of a listener, you have to provide the minimum of the full-qualified Java class name. You can provide a codebase if the CLASSPATH does not contain an entry which represents the required classes of the listener.

Also, if your listener needs some arguments, you have to provide them within the [ARGUMENTS] field.



For example, a command line of creating your own listener may look like this:

```
create listener de.ikv.grasshopper.app.explorer.Explorer
```

This command will launch a new explorer for the Agent System. One thing you have to remember if you have designed a listener which requires arguments, you have to type „“ for the codebase even no codebase specification is needed.

B.2.2 Copy

The „Copy“-command allows you to copy agent to another location. The execution state of the master agent will -in default case- be copied, i.e., the new

agent will execute the task at the same position as the master one.

However, this is just a default case. It is also possible to tell the new agent to start the task explicitly at a different location by using the `afterCopy(...)` and `afterMove(...)` methods (See the Programmer's Guide for more details).

The syntax of the „Copy“-command is:

```
copy #agent-item [<location>]
```

Same as the „Move“-command, you have to provide the item number of the agent that you want to copy. The item number can be retrieved via the „List“-command.

Beside the item number, you can optionally specify a location where to copy the Agent to. The location is made up of at least the two components `<host>/<AgencyName>` followed by an optional entry of the `<PlaceName>`.

If you ask yourself how you can tell which is the master and which ones are copies then here is the answer. Remember the 'Agent Identifier', which was described in Annex B.1.7, contains the generation information part that tells you whether the Agent is a master or a copy.

The following Figure 8 shows some examples:

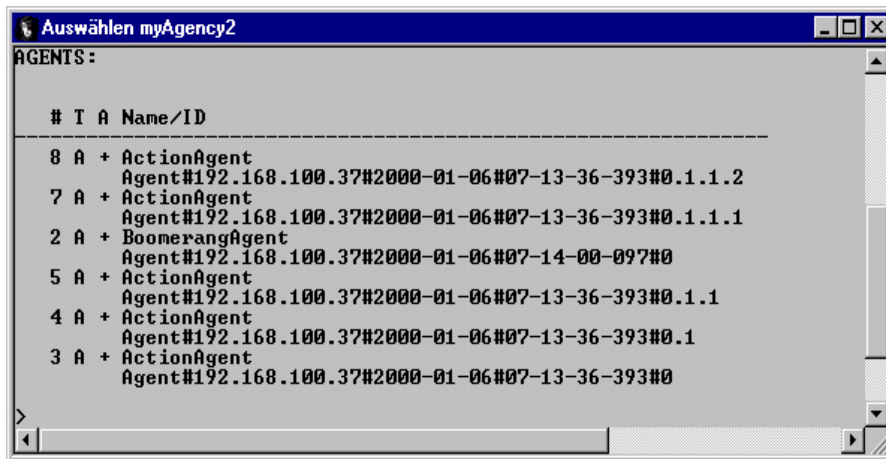


Figure 8: Copy

B.2.3 Invoke

Each Agent is associated with an action method (See the Programmer's Guide for more details). This command will invoke this. You just have to identify the Agent. This is done by specifying the item number, which can be retrieved via

„List“, of the agent.

The syntax of „Invoke“ is:

```
invoke #<agent item number>.
```

B.2.4 Move

With the „Move“-command you are able to move an agent to a particular location. The request-syntax is:

```
move #<agent-item> <location>.
```

The parameter #<agent-item> represents the item number as already mentioned in „List“-command. You can retrieve the item number of an Agent by using the „List“-command and particular under the '#'-column.

The <location> specifies the address to which the agent shall be move to. A location specification is made up of the <host>/<AgencyName>/[<PlaceName>]. A valid location, assuming #3 identifies an Agent, will for example look like this:

```
move #3 myHost/myAgency/MyPlace
```

This request will move the Agent registered under #3 from the current location to „myHost/myAgency/myPlace“.

B.2.5 Resume

This command resumes a previously suspended component. It can only apply to places and agents. In case you use this command for a component which is not suspended, an exception will be printed.

If multiple resumption of component is required, you can apply item numbers to this command. The item number can be retrieved via the „List“-command.

The syntax of the „Resume“-command is:

```
resume [#item number]*
```

Once a component is resumed, the state of the component will change from inactive to active.

B.2.6 Suspend

To stop any activities of a component, you can suspend it with this command.

The „Suspend“-command can only be applied to places and agents, which state are active. Otherwise an exception will be thrown.

Comparable with the „Resume“, you can also suspend multiple components by providing multiple item numbers.

Note, the item number of the components can be retrieved via the „List“-command.

The syntax of the „Suspend“-command is:

```
suspend [#item number]*.
```

A suspended component has the inactive state, symbolized with a '-' and which you can be seen if you use the „List“ or „Info“-command.

B.2.7 Reregister

The Reregister command enables you to reregister the Agent System to the Region Registry. Such function is required, for example, if you have to re-launch the Region Registry.

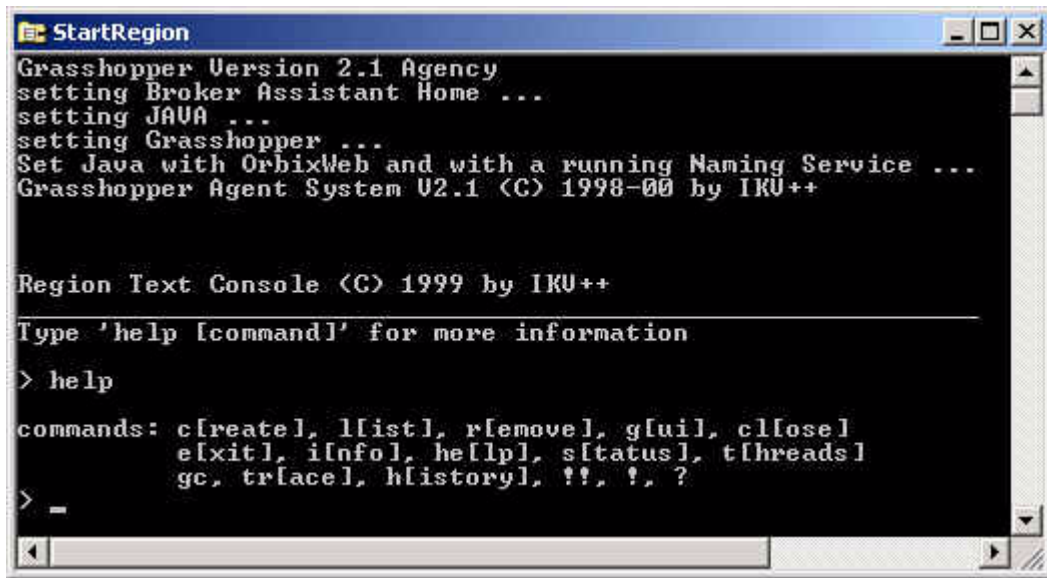
The command has no arguments.

B.3 Region Registry Specific Commands

The Region Registry has no specific commands. The only divergence to the described commands in this Chapter is the *Create* command. This provides only the feature of creating a listener (see Annex B.2.1 for how to create listeners). The creation of Agents and places doesn't make any sense at all.

All other supported commands are already covered within Annex B.1.

The following shows all commands related to a Region.



```
StartRegion
Grasshopper Version 2.1 Agency
setting Broker Assistant Home ...
setting JAVA ...
setting Grasshopper ...
Set Java with OrbixWeb and with a running Naming Service ...
Grasshopper Agent System V2.1 (C) 1998-00 by IKU++

Region Text Console (C) 1999 by IKU++

Type 'help [command]' for more information

> help

commands: c[reate], l[ist], r[emove], g[ui], c[lose]
           e[xit], i[nfo], h[elp], s[tatus], t[hreads]
           gc, tr[ace], h[istory], !!, !, ?

> _
```

Figure 9: Provided Region-Tui Commands

C Grasshopper StubGen Switches

Why do you have to know the usage of the stub generator of Grasshopper?

Well, only if you want to program your own agents that acts as a server, i.e., providing services to other client applications. In addition to this, if you are also using a Java JDK Version prior to Java 1.3 as the programming language, you have to create proxy classes, enabling other clients be able to access your service¹. For this purpose, Grasshopper provides a *stub generator*, realized in terms of a batch/shell script named `Stubgen`, which can be found in the `Bin` directory.

Note that Grasshopper provides two scripts, one for Window-based system, ending with the `*.bat` extension and `*.sh` for Solaris system.

Behavior of Stubgen

Earlier releases of Grasshopper required the *class of the server agent* as input, and the generated proxy contained all public methods of this server class. The introduction of server interfaces has been realized in Grasshopper 2.0 in order to enable agent programmers to distinguish between agent-internal public methods and those public methods that are to be accessible via the communication service.

In general, the stub generator takes an interface class file as an input and will produce a Java source file with the same name as the server interface, suffixed with the letter 'P', indicating that this represents a proxy type of class. Before a client can use this proxy, it has to be compiled and made available.

More detail information for handling a proxy of an interface is provided in the Programmer's Guide, thus refer to that handbook if you want more information.

Command-Lines of Stubgen

The shell script `Stubgen` is actually calling a Java class. You can provide information or preferences to the program via command-line arguments. The following arguments, specified using EBNF, are possible:

- (1) `<Stubgen> ::= 'Stubgen' {'-h' | '-?' | '--help'} | {<options> <classname>}`
- (2) `<classname> ::= CLASSNAME// fully qualified class name`
- (3) `<options> ::= <classpath> <output> <compilationFlags>`

1. Fortunately, Java 1.3 can automatically generate proxy classes during run-time which makes the usage of `Stubgen` obsolete.

- (4) <classpath> ::= " | {'-classpath' CLASSPATH}
- (5) <output> ::= " | {'-d' DIRECTORYNAME}
- (6) <compilationFlags> ::= " | {'--compile' '--no_source'}

Description of Each Command-line Options

The following Table 7 summarized the purpose of each command-line applicable to Stubgen.

| Command | Description |
|-------------------|--|
| CLASSNAME | The CLASSNAME represents the fully qualified Java interface class name from which a proxy is required. Note that you have to omit the extension *.class, the same as if you use the java-command for starting Java-based applications. The stubgen will take this class-file as an input and will generate the respective proxy Java-file, suffixed with a 'P' so one can distinguish them from the normal Java interface. For example, once you call: 'stubgen de.ikv.example.IDummy' a respective proxy file: .\IDummyP.java will be created at the current directory. |
| -h -? --help | If you apply this option to Stubgen, a list of all possible command-lines will be displayed on the screen. Once applied, other options will be ignored by Stubgen. |
| -classpath STRING | With this option, you can provide additional classpaths -besides those specified in the CLASSPATH environment parameter- to be used by Stubgen. |

Table 7: Command-line Options to Stubgen

| Command | Description |
|--------------------------|---|
| <code>-d STRING</code> | If you want the generated stub to be written out to a particular directory, you can specify this via this option. The value of <code>STRING</code> must contain valid and existing directory path name. |
| <code>--compile</code> | This option will automatically compile the newly created proxy Java file. The output, i.e., the class-file will be stored in the current directory if not use in conjunction with <code>'-d'</code> option. |
| <code>--no_source</code> | This option can only be used together with <code>'--compile'</code> option. Once applied, no source file of the proxy will be output. |

Table 7: Command-line Options to Stubgen

D Troubleshooting

Multiple JDK Installation

If multiple JDK version are installed on your system, Grasshopper will take the version which appears first within the PATH environment parameter.

For example, if you have installed JDK 1.2 and JDK 1.3 and PATH has the respective entries, such as `.../jdk12/bin/java.exe;.../jdk13/bin/java.exe; ...`, Grasshopper will use JDK 1.2.

In order to switch to JDK 1.3, you just have to reorder the PATH entries.

E Acronyms

| | |
|-------|---|
| ADS | Agency Domain Service |
| API | Application Programming Interface |
| CORBA | Common Object Request Broker Architecture |
| FIPA | Foundation of Intelligent Physical Agents |
| GUI | Graphical User Interface |
| IIOB | Internet Inter-ORB Protocol |
| IOR | Interoperable Reference |
| JDK | Java Development Kit |
| JRE | Java Runtime Environment |
| JVM | Java Virtual Machine |
| LDAP | Lightweight Directory Access Protocol |
| MA | Mobile Agent |
| MASIF | Mobile Agent System Interoperability Facility |
| OMG | Object Management Group |
| ORB | Object Request Broker |
| RMI | Remote Method Invocation |
| SSL | Secure Socket Layer |
| TUI | Textual User Interface |
| UI | User Interface |

F Index

A

agent system
 exit 56
 explore 57

C

catalog
 close 68, 71
 copy 72
 create 67
 load 68
 preferences 67
 save 67
 save as 68
 toolbar 69, 72
console
 preferences 71

E

explorer
 close 61
 copy 62
 copy to 64
 cut 61
 delete 62
 exit 61
 help 66
 invoke 65
 menu bar 58
 move to 64
 new 63
 paste 62
 preferences 61
 properties 65
 property view 59, 77
 refresh 62
 region registration 77
 resume 64
 status bar 62
 statusbar 59
 suspend 64
 toolbar 58, 62, 73
 tools 65

tree view 59, 75

R

region registration
 exit 57
 explore 57

S

start
 agent system 56
 region registration 55

