Software Engineering Project (2IP40)

Project Group 1

Architectural Design Document

version 1.0.0 (Approved), 27 April 2006

Project Team:	Sven Bego	0550191
	Roel Coset	0548132
	Robert Leeuwestein	0546746
	Maarten Leijten	0547649
	Ivo van der Linden	0547632
	Joery Mens	0547515
	Marcel Moreaux	0499480
	Tim Muller	0547961
Project Manager:	Tom Kleijkers	0515015
Senior Manager:	L. Somers	TU/e HG 7.83
Advisor:	Y.Usenko	TU/e HG 5.71
Customer:	M. ter Linden	Dutch Space
	H. de Wolf	Dutch Space

Technische Informatica, Eindhoven University of Technology, Eindhoven

Abstract

This document contains the Architectural Design for the SPINGRID system. This project is one of seven assignments for the course 2IP40 Software Engineering at Eindhoven University of Technology. The architectural design was developed using the software requirements described in [SRD]. The document complies with the Architectural Design Document (ADD) from the Software Engineering Standard, as set by the European Space Agency [ESA].

Contents

Document Status Sheet

Document Title	Architectural Design Document
Document Identification	SPINGRID/Documents/Product/ADD/1.0.0
Author(s)	S. Bego, R. Coset, R. Leeuwestein, M. Leijten, I. v.d. Linden, J. Mens, M. Moreaux, T. Muller
Version	1.0.0
Document Status	draft / internally accepted / conditionally approved / [✓ approved]

Version	**Date**	**Author(s)**	**Summary**
0.0.1	14-02-2006	M. Leijten	Document creation
0.0.2	22-03-2006	S. Bego, R. Coset, R. Leeuwestein, M. Leijten, I. v.d. Linden	Draft
0.0.3	06-04-2006	S. Bego, R. Coset, R. Leeuwestein, M. Leijten, I. v.d. Linden, J. Mens, M. Moreaux	Draft
0.0.4	13-04-2006	S. Bego, R. Coset, R. Leeuwestein, M. Leijten, I. v.d. Linden, J. Mens, M. Moreaux, T. Muller	Version for first internal review
0.0.5	19-04-2006	S. Bego, R. Coset, R. Leeuwestein, M. Leijten, I. v.d. Linden, J. Mens, M. Moreaux, T. Muller	Version for second internal review
0.1.0	19-04-2006	S. Bego, R. Coset, R. Leeuwestein, M. Leijten, I. v.d. Linden, J. Mens, M. Moreaux, T. Muller	Conditionally Approved
1.0.0	27-04-2006	S. Bego, R. Coset, R. Leeuwestein, M. Leijten, I. v.d. Linden, J. Mens, M. Moreaux, T. Muller	Approved

Document Change Report

Document Title	Architectural Design Document
Document Identification	SPINGRID/Documents/Product/ADD/1.0.0
Date of Changes	N/A

Section Number	**Reason for Change**

Chapter 1

Introduction

1.1 Purpose

The Architectural Design Document (ADD) describes the basic system design for the software to be made during the SPINGRID project. It describes the physical model; this is a decomposition of the software into components. Each component is described in terms of its external interfaces and the dependencies on other components, this in order to allow the programmers in the next phase of the project to work in parallel.

1.2 Scope

The software implements a computational grid. This grid is able to execute jobs when it receives an application accompanied by a set of data files. By hiding the complexity of grid technology the system will be easy to use. Usability is also increased by offering a web-based front-end for users to access the system.

1.3 List of definitions and abbreviations

1.3.1 Definitions

Agent	Application that is used by a resource provider to retrieve and execute jobs.
Application	A non-interactive data processing application consisting of executables, scripts and/or auxiliary data files that reads one or more input data files and writes one ore more output files.
Application Provider	An application provider can offer a set of applications to the SPINGRID system. They can restrict access for projects and for resource providers to their applications.
Client	Application that is used by all the users except the resource provider who uses the agent application.
Computational Grid	A hardware and software infrastructure that enables coordinated resource sharing within dynamic organizations consisting of individuals, institutions and resources.
Customer	Dutch Space B.V.
Dispatcher	A dispatcher acts like a server and manages the distribution of jobs over the computational grid.
Data Provider	A data provider can offer a set of datafiles to the SPINGRID system. They can restrict access for projects and for resource providers to their datafiles.
Job	Specification of application, configuration data, input and/or output data files and scheduler specific data (priority, preferred resource, etc).
Job Provider	Job providers are users that offer a job to a project. They have to be a member of that particular project.
GRP	A generic package that is converted from a HTTP response and is formatted as two strings (status code and message).
GSP	A generic package that can be converted to a HTTP request header and is formatted as a list of pairs of two strings (variable and value).
PM	Project Manager.
Project	A collection of jobs with specified access rights to which users (project members) can be assigned.
Project Administrator	The project administrators administrate projects and can assign and remove job providers, configure a project and restrict access for resource providers.
Resource Provider	Resource providers are users that offer time on their computers to the SPINGRID system. They can restrict access to their computer for application providers and projects.
Role	The actions and activities assigned to a person.
SPINGRID	A computational grid using SPINGRID software.
SPINGRID Software	Software developed by Dutch Space and TU/e to build computational grids for distributed data processing.
SPINGRID System	The full name of the entire system.
System Administrator	The system administrator oversees the entire SPINGRID system and has the right to configure the system, to create and remove projects and assign and remove project administrators.

1.3.2 Abbreviations

ADD	Architectural Design Document
ESA	European Space Agency
DDD	Detailed Design Document
GRP	Generic Receive Package
GSP	Generic Send Package
HTTP	Hypertext Transfer Protocol
JRE	Java Runtime Environment
OMT	Object Modeling Technique
SRD	Software Requirements Document
UML	Unified Modeling Language

1.4 Documents

1.4.1 Reference Documents

[ESA]	*ESA Software Engineering Standards (ESA PSS-05-0 Issue 2)*, ESA Board for Software Standardization and Control (BSSC), 1991
[JSDL]	*Job Submission Description Language (JSDL) Specification*, Version 1.0, November 2005
[DDD]	*Detailed Design Document*, SPINGRID team, TU/e, not yet available
[UML]	*Practical UML: A Hands-On Introduction for Developers*, Togethersoft, 2000

1.4.2 Applicable Documents

[URD]	*User Requirements Document*, SPINGRID team, TU/e, version 1.0.0, February 2006
[SRD]	*Software Requirements Document*, SPINGRID team, TU/e, version 1.0.1, March 2006

1.5 Overview

Chapter 2 of this document is a short introduction to the general context of the SPINGRID system to be made and the background of the project. It also discusses the systems design decisions. In chapter 3 the relationship of the SPINGRID system with external systems is described. Chapter 4 describes the decomposition of the system in subcomponents and the relations between them. Each of these subcomponents is specified in detail in chapter 5. Chapter 6 gives an estimate of the resources needed to develop and maintain the SPINGRID system. The last chapter, chapter 7, provides two traceability matrices. One matrix traces each Software Requirement (SR) to a part of the architectural design described in chapter 5, the other one does the reverse.

Appendix A describes the protocol implemented by the applications to communicate with each other. Appendix B explains the user interface (the command line syntax).

Chapter 2

System overview

2.1 Background

The function and purpose of this project are described in [URD, sections 2.1, 2.2 and 2.4]. The environment in which the system the system runs is described in [URD, section 2.5].

2.2 Basic design and context

As already decided in [SRD], the SPINGRID system will consist of three components on the highest level. These three components will be implemented as three applications. These components are called agent, dispatcher and client and the relations between them and with the users can be seen in figure 2.1.

The agent is used by the resource provider. All resource providers requirements, as described in the [URD, section 4.4], are fulfilled by the agent. An agent is connected to one or more independent dispatchers.

The client is used by the application, data and job providers. All application, data and job providers requirements, as described in the [URD, sections 4.6, 4.7 and 4.8], are fulfilled by the client. The system admin and project admin also use the client and these requirements ([URD sections 4.3 and 4.5]) are also fulfilled by the client. A client is connected to one dispatcher. The main difference and therefore the reason to make a agent and client, is that the agent is communicating with the dispatcher on regular basis, this is not the case with a client. Communication between the client and dispatcher only exists when a commando is given by the user, for example submitting a job.

The dispatcher acts like a server and manages the distribution of jobs over the computational grid. A dispatcher is connected to zero or more agents and to zero or more clients. Dispatchers operate independently.

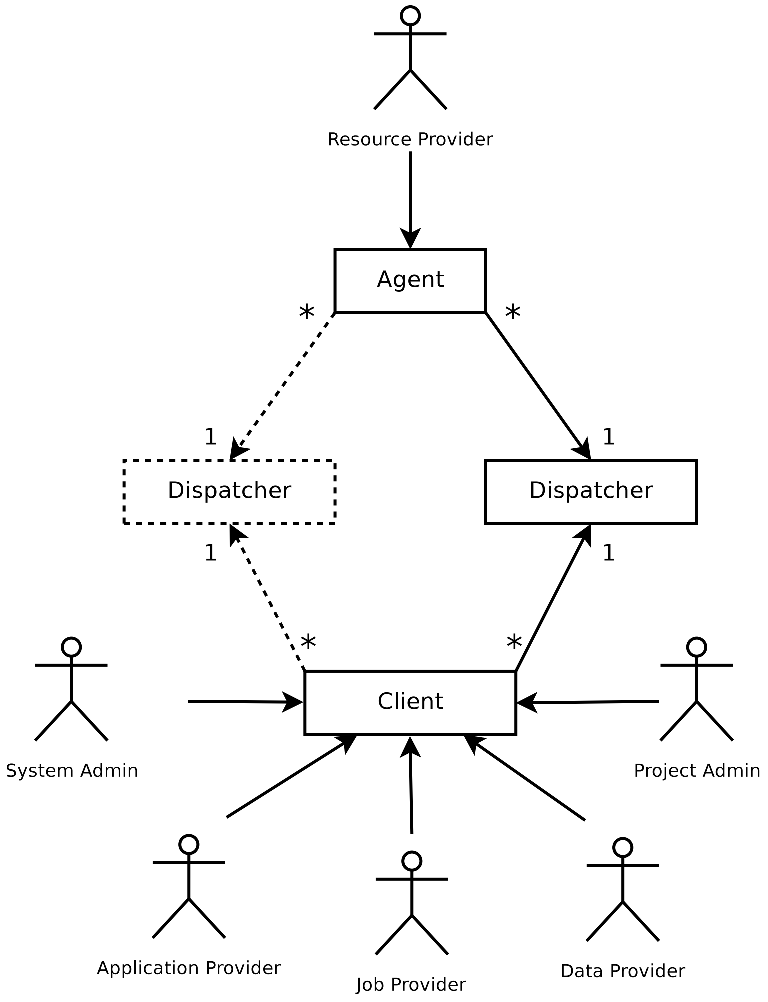

Figure 2.1: High Level Model

[SRD] describes how the user requirements are transformed into a logical model. This logical model mainly describes the relation between the different entities in the product. After the software requirements and the logical model are transformed into an architectural model, this architectural model can be transformed into a detailed design which is done in [DDD].

2.3 Design decisions and preliminary information

2.3.1 Java Runtime Environment

As stated in the User Requirements Document [URD] the design decision was made to implement all software parts of the SPINGRID system in Java. The main reason for this is the platform independency that Java provides. This platform independency enables the SPINGRID software to be run on a large variety of computers, allowing more computers to participate in the grid. Other considerations were maintainability and the high availability of Java implementations.

The SPINGRID software will require Sun JRE 1.4.2 or 1.5.0.

2.3.2 MySQL in combination with JDBC

The SPINGRID dispatcher program needs a relational database to store its data. For this purpose, it will use MySQL as the database, and it will use JDBC, with the MySQL Connector/J driver, to access the database. This requires the java.sql package, which is supplied with Sun J2SE 1.4.2, and Sun J2SE 1.5.0.

Additionally, the MySQL Connector/J JDBC driver is required, which is available from the MySQL website[1]. In Debian GNU/Linux, this is available in the libmysql-java package.

2.3.3 XML Parser

For parsing XML, the SPINGRID software will use JAXP, specifically the javax.xml.parsers. DocumentBuilder class. This requires the javax.xml.parsers, org.xml.sax, and the org.w3c.dom packages, all of which are supplied with Sun J2SE 1.4.2 and Sun J2SE 1.5.0.

2.3.4 HTTP Message

The HTTP Message component can build and parse HTTP Messages. A component using the HTTP Message component can send HTTP Messages over the network without knowing the exact layout of such a HTTP Message. Note that the HTTP Message component will be build by the SPINGRID team.

[1] `http://dev.mysql.com/downloads/connector/j/3.1.html`

Chapter 3

System context

90 The environment in which the SPINGRID system runs is generally described in [URD, section 2.5] and [SRD, section 2.4].

Chapter 4

System design

4.1 Design Method

The method used to design the component model is the Object Oriented design method OMT. The present model is a decomposition view into components and is made with [UML].

4.2 Decomposition description

4.2.1 Dispatcher

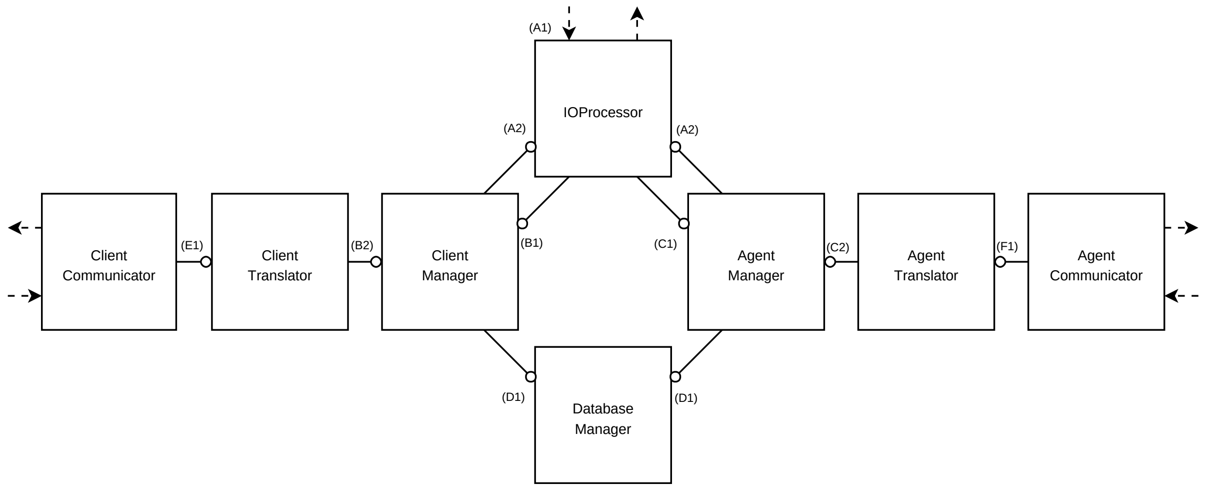

Figure 4.1: Components of the Dispatcher-software

Legend of figure 4.1:

- (A1): IDInputProcessor
- (A2): IDOutputProcessor

- (B1): IDCCommandInput
- (B2): IDUserCommands, IDProjectAdminCommands, IDSystemAdminCommands, IDJobProviderCommands, IDApplicationProviderCommands, IDDataProviderCommands
- (D1): IDQueryHandler
- (C1): IDACommandInput
- (C2): IDResourceProviderCommands
- (E1): IDCPackageInput
- (F1): IDAPackageInput

The components and interfaces of the dispatcher application are displayed in figure 4.1. The component and interfaces descriptions can be found in section 5.2.

The dispatcher application will use several threads. Firstly, every time a command is received, instances of ClientTranslator/AgentTranslator ClientManager/AgentManager will be created in independent threads, thus allowing several commands to be performed at the same time. The other components can run in one thread together.

IOProcessor

This component of the dispatcher takes input from and gives output to the user of the software. However, there will not be much input from the user as the dispatcher is mostly remote-controlled (by the System Admin). One input command could be *shutdown*. The output will mostly consist of logs.

DB

The database-component is responsible for communication with the SQL-database. It receives SQL-queries as input from the ClientManager and AgentManager and returns the results (as a set of records in case of a SELECT-query).

ClientManager

The ClientManager receives commands from ClientTranslator and verifies the access rights of the user that sent the command. It interacts with the database to gather data that was requested and to update the desired settings.

ClientTranslator

When a method in the ClientTranslator is called, it translates the parameters of the method into a GSP. The GSPs are then send to the ClientCommunicator. GRPs from the Client-Communicator are evaluated so that the appropriate methods in the ClientManager can be called.

ClientCommunicator

This component takes care of the communication with the Client using HTTP.

AgentManager

The AgentManager receives messages and commands from AgentTranslator. The messages can be either a request for a job, a 'working-update' and a 'job-completed' message. The commands are used to perform the desired actions. The AgentManager contacts the database to retrieve and update information about jobs and change settings.

AgentTranslator

When a method in the AgentTranslator is called, it translates the parameters of the method into a GSP. The GSPs are then send to the AgentCommunicator. GRPs from the Agent-Communicator are evaluated so that the appropriate methods in the AgentManager can be called.

AgentCommunicator

This component takes care of the communication with the Agent using HTTP.

4.2.2 Client

The components and interfaces of the client application are displayed in figure 4.2. The component and interfaces descriptions can be found in section 5.3.

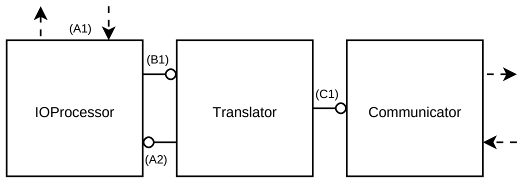

Figure 4.2: Components of the Client-software

Legend of figure 4.2:

- (A1): ICInputProcessor
- (A2): ICOutputProcessor
- (B1): ICUserCommands, ICProjectAdminCommands, ICSystemAdminCommands, ICJobProvider-Commands, ICApplicationProviderCommands, ICDataProviderCommands
- (C1): ICCommunicator

All of the components in the client application will run in a single thread.

IOProcessor

This component of the client is responsible for processing the input of the user (this can be a human being, but also an automated process generating input for the client). If the input is valid, the component calls the corresponding method in the Translator component. Otherwise, it will let the user know the input is invalid. Furthermore, other components can tell this component to output messages to the user.

Translator

When a method in the Translator is called, it translates the parameters of the method into a GSP. The GSPs are then send to the Communicator. GRPs from the Communicator are evaluated so that the appropriate message can be send to the user through the IOProcessor.

Communicator

This component is responsible for implementing the communication protocol described in appendix A.1. Its main job is to translate GSPs received from the Translator into messages that comply to the protocol and send those to the dispatcher. Messages received from the dispatcher are translated into GRPs and are send to the Translator.

4.2.3 Agent

The components and interfaces of the agent application are displayed in figure 4.3. The component and interfaces descriptions can be found in section 5.2.

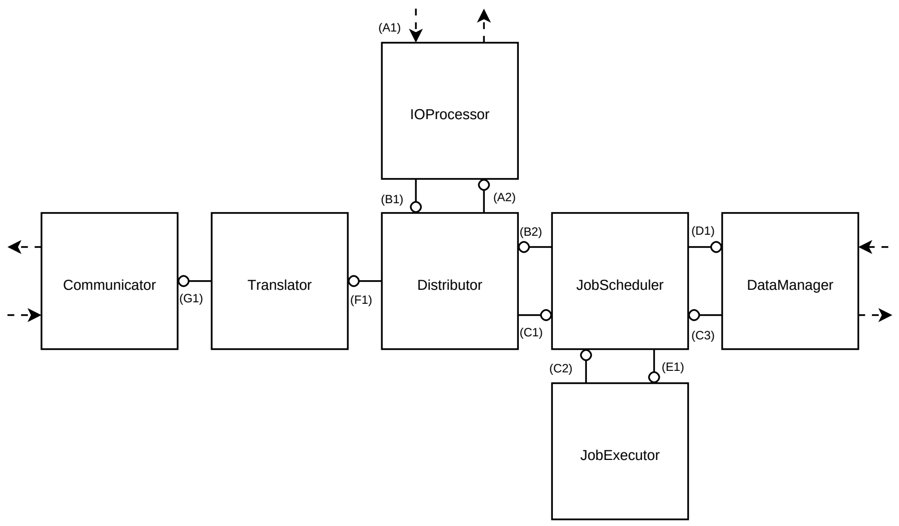

Figure 4.3: Components of the Agent-software

Legend of figure 4.3:

- (A1): IAInputProcessor

- (A2): IAOutputProcessor
- (B1): IADistributorForIOProcessor
- (B2): IADistributorForJobScheduler
- (C1): IAJobSchedulerForDistributor
- (C2): IAJobSchedulerForJobExecutor
- (C3): IAJobSchedulerForDataManager
- (D1): IADataManager
- (E1): IAJobExecutor
- (F1): IAResourceProviderCommands
- (G1): IACommunicator

The components of the agent application will run in several threads. The Translator and Communicator will be in the same thread. All the other components will run in another thread. For every job that will be executed, another thread will be created.

IOProcessor

The IOProcessor manages the input received from and output given to the user. It redirects input to and receives output from the Distributor.

Distributor

The Distributor receives tasks from the IOProcessor and the JobScheduler and passes it to the Translator. This component is seperated from the Translator for multi-threading reasons.

JobScheduler

The JobScheduler is the main component of the agent. It asks the Communicator (through the Distributor and the Translator) to ask the dispatcher for a job. If a job is received, the DataManager is asked to download the necessary data. When this is done, the JobExecutor will launch the job process. After the job has been completed, the DataManager is asked to upload the results. The Communicator component will inform the dispatcher that the job has been completed.

JobExecutor

The JobExecutor is the component which will execute the job.

DataManager

The DataManager will download data needed to perform jobs and will upload results when jobs are completed.

Translator

When a method in the Translator is called, it translates the parameters of the method into a GSP. The GSPs are then send to the Communicator. GRPs from the Communicator are evaluated so that the appropriate message can be send to the user through the Distributor.

Communicator

The Communicator component parses the configuration file that describes the characteristics of the machine on which the Agent is running. When the component polls the dispatcher for jobs, it sends this information to the dispatcher. Furthermore, the Communicator component will inform the dispatcher of the status of the Agent.

Chapter 5

Component description

All the components in the SPINGRID system are described in this chapter. Note that every method can raise an exception if something goes wrong. An example of this can be if the user is calling the method and the user is not authorized to call this method with the given arguments. The different types of exceptions are described in [DDD]. For now we will say that all methods can raise zero or more exceptions which are all a subclass of SPINGRIDException.

5.1 General Components

5.1.1 Logger

Every application should contain at least the Logger component. This component should be tight to every other component that wants to log actions to the log files. For clarity, this component has not been put in figures 4.1, 4.2 and 4.3.

Type The Logger is a component in the SPINGRID project and is part of all applications.

Purpose Logger implements software requirements SR_9160.

Function The function of this component is to log the actions of the applications. It is capable of logging regular messages and error messages separately.

Subordinates This component consists of a set of classes.

Dependencies None.

Interfaces

1. ILogger

- `Log(String msg)`
  This method should be called when an important regular event has happened that needs to be logged in a log file. This method then appends the give string to the log file.

- `Error(String msg)`
  This method should be called when an error message needs to be logged. The error message is appended to the standard error log file.

5.1.2 HTTP Message

Every application should contain at least the HTTP Message component. This component is a subcomponent of all components that handle communication. On the requesting side as well as the responding side.

Type HTTP Message is a component in the SPINGRID project and is part of all applications. It is always a subcomponent of a communication component.

Purpose HTTP Message implements software requirements SR_2010, SR_2011, SR_2020, SR_2021, SR_2030, SR_2031, SR_2040, SR_2041, SR_2060, SR_2070, SR_2080, SR_2090, SR_2100, SR_3010, SR_3020, SR_3021, SR_3030, SR_3031, SR_3040, SR_3050, SR_3051, SR_3060, SR_3061, SR_3070, SR_3071, SR_4010, SR_4011, SR_4020, SR_4021, SR_4030, SR_4031, SR_4032, SR_4033, SR_4040, SR_4041, SR_4042, SR_4043, SR_4050, SR_4051, SR_4060, SR_4070, SR_5010, SR_5011, SR_5020, SR_5021, SR_5030, SR_5040, SR_5050, SR_5070, SR_5071, SR_6010, SR_6011, SR_6020, SR_6021, SR_6030, SR_6040, SR_6050, SR_6060, SR_6061, SR_7010, SR_7011, SR_7012, SR_7013, SR_7020, SR_7030, SR_7040, SR_8000 and SR_9130.

Function Increase maintainability and make it easier to update to a newer HTTP version.

Subordinates This component consists of a set of classes.

Dependencies None.

Interfaces

1. IHTTPRequest

   - `HTTPRequest build(String HTTPVersion, String method, String URL, String content)`
     This method should be called by clients and agents to create a HTTPRequest object.
     Results:

  - An HTTPRequest object. Its fields get the value of the corresponding parameters. Content will also be parsed to a list of variables.

- `HTTPRequest parse(BufferedReader is)`
  This method should be called by the dispatcher to create a HTTPRequest object.
  Results:
  - An HTTPRequest object. Its fields get the value that can be parsed from the incoming message is. Content will also be parsed to a list of variables.

- `String getHTTPVersion()`
  This method should be called if a component wants to know the HTTP version is used.
  Results:
  - HTTPVersion field

- `String getMethod()`
  This method should be called if a component wants to know which method is used in the request.
  Results:
  - Method field

- `String getURL()`
  This method should be called if a component wants to know which URL is requested.
  Results:
  - URL field

- `String getContent()`
  This method should be called get information from the data in the message.
  Results:
  - Content field

- `String[] getValues(String key)`
  This method should be called to get information over the message.
  Results:
  - If attribute with given key exists: return all the values associated with it.
  - If attribute with given key does not exist: return null.

- `String getAValue(String key)`
  This method should be called to get information over the message.
  Results:
  - If attribute with given key exists: return a value associated with it.
  - If attribute with given key does not exist: return null.

- `String getVariable(String key)`
  This method should be called to get information from the data in the message.
  Results:
  - If variable with given key exists: return the value associated with it.
  - If variable with given key does not exist: return null.

- `addValue(String key, String value)`
  This method should be called to add information about the network to the HTTPRequest. If there is no attribute named key, an attribute key with value is created. If this key exist, value will be added.

- `addVariable(String key, String value)`
  This method should be called to add data to the HTTPRequest. If there is no variable named key, a variable key with value is created. If this key exist, it's actions are undefined.

- `byte[] getBytes()`
  This should be called when sending data over the network. E.g. socket.send(myHttp Request.getBytes())
  Results:
  - A HTTP message, in the form of a sequence of bytes. This can be sent over the network.

2. IHTTPResponse

- `Static HTTPResponse build(String HTTPVersion, int codeN, String codeS, String content)`
  This method should be called by the dispatcher to create a HTTPResponse object.
  Results:
  - An HTTPResponse object. Its fields get the value of the corresponding parameter.

- `Static HTTPRequest parse(BufferedReader is)`
  This method should be called by clients and agents to create a HTTPRequest object.
  Results:
  - An HTTPResponse object. Its fields get the value that can be parsed from the incoming message is.

- `String getHTTPVersion()`
  This method should be called if a component wants to know the HTTP version is used.
  Results:
  - HTTPVersion field

- `String getCodeN()`
  This method should be called if a component wants to know the statuscode.
  Results:
  - Statuscode number field

- `String getCodeS()`
  This method should be called if a component wants to know the statusname.
  Results:
  - Statusname field

- `String getContent()`
  This method should be called if a component want the content in the response.
  Results:
  - Content field
- `String[] getValues(String key)`
  This method should be called to get information over the message.
  Results:
  - If attribute with given key exists: return all the values associated with it.
  - If attribute with given key does not exist: return null.
- `String getAValue(String key)`
  This method should be called to get information over the message.
  Results:
  - If attribute with given key exists: return a value associated with it.
  - If attribute with given key does not exist: return null.
- `addValue(String key, String value)`
  This method should be called to add information about the network to the HTTPRequest. If there is no attribute named key, an attribute key with value is created. If this key exist, value will be added.
- `byte[] getBytes()`
  This should be called when sending data over the network. E.g. socket.send(myHttp Request.getBytes())
  Results:
  - A HTTP message, in the form of a sequence of bytes. This can be sent over the network.

5.1.3 JSDL Description

This component deals with parsing JSDL descriptions. It is only used by the Dispatcher and Agent applications.

Type JSDL Description is a component in the SPINGRID project and is part of most applications.

Purpose JSDL Description implements software requirements SR_1210, SR_1220, SR_1230, SR_1240, SR_1310, SR_1320, SR_1330, SR_1410, SR_1411, SR_1412, SR_1413, SR_1414, SR_1415, SR_1416, SR_1417, SR_1418, SR_1419, SR_1420, SR_1421, SR_1422, SR_1423, SR_1424, SR_1425, SR_1426, SR_1427, SR_1510, SR_1520, SR_1530, SR_1540, SR_1550, SR_1560 and SR_7140.

Function The function of this component is to parse a JSDL Job Description, extracting the information embedded inside it.

Subordinates This component consists of a set of classes.

Dependencies JAXP XML parser.

Interfaces

(a) IJSDLParser

- `String GetJobName(JSDL jobdesc)`
  This method should be called to retrieve the job's name from a JSDL job description.
  Results:
  - A string containing the job's name.
- `String GetJobDescription(JSDL jobdesc)`
  This method should be called to retrieve the job's description from a JSDL job description.
  Results:
  - A string containing the job's description.
- `String[] GetJobAnnotations(JSDL jobdesc)`
  This method should be called to retrieve the job's annotation(s) from a JSDL job description.
  Results:
  - An array of strings containing the job's annotation(s).
- `String[] GetJobProjects(JSDL jobdesc)`
  This method should be called to retrieve the job's project name(s) from a JSDL job description.
  Results:
  - An array of strings containing the job's project name(s).
- `String GetApplicationName(JSDL jobdesc)`
  This method should be called to retrieve the job application's name from a JSDL job description.
  Results:
  - A string containing the job application's name.
- `String GetApplicationVersion(JSDL jobdesc)`
  This method should be called to retrieve the job application's version from a JSDL job description.
  Results:
  - A string containing the job application's version.
- `String GetApplicationDescription(JSDL jobdesc)`
  This method should be called to retrieve the job application's description from a JSDL job description.
  Results:
  - A string containing the job application's description.

- `CandidateHosts GetCandidateHosts(JSDL jobdesc)`
  This method should be called to retrieve the candidate hosts for running this job from a JSDL job description.
  Results:
  - A candidatehosts containing the hosts that are eligible for running this job on.

- `FileSystem[] GetFileSystems(JSDL jobdesc)`
  This method should be called to retrieve the filesystem definition(s) from a JSDL job description.
  Results:
  - An array of filesystems containing the job's filesystem definition(s).

- `Boolean GetExclusiveExecution(JSDL jobdesc)`
  This method should be called to retrieve the exclusive execution flag from a JSDL job description.
  Results:
  - A boolean, representing whether or not the job should have exclusive execution.

- `OperatingSystem GetOperatingSystem(JSDL jobdesc)`
  This method should be called to retrieve the allowed operating system(s) from a JSDL job description.
  Results:
  - An operatingsystem containing specifications for the allowed operating system(s) for this job.

- `String GetCPUArchitecture(JSDL jobdesc)`
  This method should be called to retrieve the job's target CPU architecture from a JSDL job description.
  Results:
  - A string containing the name of the job's target CPU architecture.

- `Range GetICPUSpeed(JSDL jobdesc)`
  This method should be called to retrieve the job's required individual CPU speed from a JSDL job description.
  Results:
  - A range specifying the allowed values for the individual CPU speed.

- `Range GetICPUTime(JSDL jobdesc)`
  This method should be called to retrieve the job's required individual CPU time from a JSDL job description.
  Results:
  - A range specifying the allowed values for the individual CPU time.

- `Range GetICPUCount(JSDL jobdesc)`
  This method should be called to retrieve the job's required individual CPU count from a JSDL job description.
  Results:
  - A range specifying the allowed values for the individual CPU count.

- `Range GetINetworkBandwidth(JSDL jobdesc)`
  This method should be called to retrieve the job's required individual network bandwidth from a JSDL job description.
  Results:
  - A range specifying the allowed values for the individual network bandwidth.
- `Range GetIPhysicalMemory(JSDL jobdesc)`
  This method should be called to retrieve the job's required individual physical memory from a JSDL job description.
  Results:
  - A range specifying the allowed values for the individual physical memory.
- `Range GetIVirtualMemory(JSDL jobdesc)`
  This method should be called to retrieve the job's required individual virtual memory from a JSDL job description.
  Results:
  - A range specifying the allowed values for the individual virtual memory.
- `Range GetIDiskSpace(JSDL jobdesc)`
  This method should be called to retrieve the job's required individual disk space from a JSDL job description.
  Results:
  - A range specifying the allowed values for the individual disk space.
- `Range GetTCPUTime(JSDL jobdesc)`
  This method should be called to retrieve the job's required total CPU time from a JSDL job description.
  Results:
  - A range specifying the allowed values for the total CPU time.
- `Range GetTCPUCount(JSDL jobdesc)`
  This method should be called to retrieve the job's required total CPU count from a JSDL job description.
  Results:
  - A range specifying the allowed values for the total CPU count.
- `Range GetTPhysicalMemory(JSDL jobdesc)`
  This method should be called to retrieve the job's required total physical memory from a JSDL job description.
  Results:
  - A range specifying the allowed values for the total physical memory.
- `Range GetTVirtualMemory(JSDL jobdesc)`
  This method should be called to retrieve the job's required total virtual memory from a JSDL job description.
  Results:
  - A range specifying the allowed values for the total virtual memory.
- `Range GetTDiskSpace(JSDL jobdesc)`
  This method should be called to retrieve the job's required total disk space from a JSDL job description.
  Results:

  - A range specifying the allowed values for the total disk space.

- `Range GetTResourceCount(JSDL jobdesc)`
  This method should be called to retrieve the job's required total resource count from a JSDL job description.
  Results:
  - A range specifying the allowed values for the total resource count.
- `DataStaging[] GetDataStaging(JSDL jobdesc)`
  This method should be called to retrieve the datastaging specification(s) from a JSDL job description.
  Results:
  - An array of datastagings containing the datastaging specification(s) for this job.

5.2 Dispatcher Components

5.2.1 IOProcessor

Type The IOProcessor is a component in the dispatcher application of the SPINGRID project.

Purpose IOProcessor implements software requirements SR_8000 and SR_9030.

Function This component has the function to process inputs from the outside world and process outputs from the other components of the dispatcher.

Subordinates This component consists of a set of classes.

Dependencies None.

Interfaces

1. IDInputProcessor (see (A1) in figure 4.1)
   - `Input(String[] args)`
     This method should be called when the outside world has a sequence of characters ready to process. This sequence will be stored in the args variable.
2. IDOutputProcessor (see (A2) in figure 4.1)

- `Output(String str)`
  This method should be called when another component wants to send a sequence of characters to standard output. This sequence will be printed in the console.

5.2.2 ClientCommunicator

Type The ClientCommunicator is a component in the dispatcher application of the SPINGRID project.

Purpose ClientCommunicator implements software requirements SR_2010, SR_2011 SR_2020, SR_2021, SR_2030, SR_2031, SR_2040 SR_2041, SR_2060, SR_2070, SR_2080, SR_2090, SR_2100, SR_4010, SR_4011, SR_4020, SR_4021, SR_4030, SR_4031 SR_4032, SR_4033, SR_4040, SR_4041, SR_4042, SR_4043, SR_4050, SR_4051, SR_4060, SR_4070, SR_5010, SR_5011, SR_5020, SR_5021, SR_5030, SR_5040, SR_5050, SR_5070, SR_5071, SR_6010, SR_6011, SR_6020, SR_6021, SR_6030, SR_6040, SR_6050, SR_6060, SR_6061, SR_7010, SR_7011, SR_7012, SR_7013, SR_7020, SR_7030, SR_7040 and SR_8000.

Function This component has the function to communicate with the client. This is one of the two components in the dispatcher application able to communicate with the outside world.

Subordinates This component consist of a set of classes:

1. Receiver
   Contains a loop that receives HTTP requests from a client and creates instances of a communication class.

2. Communication
   Handles the request, uses a blocking method to get the result from the dispatcher and sends this back to the client.

Dependencies HTTP Message component.

Interfaces None.

5.2.3 ClientTranslator

Type The ClientTranslator is a component in the SPINGRID project and is part of the dispatcher application.

Purpose ClientTranslator implements software requirements SR_2010, SR_2011 SR_2020, SR_2021, SR_2030, SR_2031, SR_2040 SR_2041, SR_2060, SR_2070, SR_2080, SR_2090, SR_2100, SR_4010, SR_4011, SR_4020, SR_4021, SR_4030, SR_4031 SR_4032, SR_4033, SR_4040, SR_4041, SR_4042, SR_4043, SR_4050, SR_4051, SR_4060, SR_4070, SR_5010, SR_5011, SR_5020, SR_5021, SR_5030, SR_5040, SR_5050, SR_5070, SR_5071, SR_6010, SR_6011, SR_6020, SR_6021, SR_6030, SR_6040, SR_6050, SR_6060, SR_6061, SR_7010, SR_7011, SR_7012, SR_7013, SR_7020, SR_7030, SR_7040 and SR_8000.

Function The ClientTranslator has several functions:

- Receive GRP's from the ClientCommunicator.
- Send commands to the ClientManager.
- Translate commands to generic packages and generic packages to commands.

Subordinates This component consists of a set of classes.

Dependencies None.

Interface

1. IDCPackageInput (see (E1) in figure 4.1)
    - `GRP PerformCommand(GSP Package)`
      This method should be called when a GRP needs to be translated into a command. This command is then executed.
      Results:
        - The result package of a command.

5.2.4 ClientManager

The ClientManager receives commands from ClientTranslator and verifies the access rights of the user that sent the command. It interacts with the database to gather data that was requested.

The verification of the acces rights has to be done in every method listed below. This implies that the method needs some kind of identification of the user to verify this. The identification will be provided to the method as an argument. These arguments are left out of this document to keep the methods less complex. The identification will be further explained in the the [DDD].

Type The ClientManager is a component in the SPINGRID project and is part of the dispatcher application.

Purpose ClientManager implements software requirements SR_2010, SR_2011 SR_2020, SR_2021, SR_2030, SR_2031, SR_2040 SR_2041, SR_2050, SR_2060, SR_2070, SR_2080, SR_2090, SR_2100, SR_4010, SR_4011, SR_4020, SR_4021, SR_4030, SR_4031 SR_4032, SR_4033, SR_4040, SR_4041, SR_4042, SR_4043, SR_4050, SR_4051, SR_4060, SR_4070, SR_5010, SR_5011, SR_5020, SR_5021, SR_5030, SR_5040, SR_5050, SR_5070, SR_5071, SR_6010, SR_6011, SR_6020, SR_6021, SR_6030, SR_6040, SR_6050, SR_6060, SR_6061, SR_7010, SR_7011, SR_7012, SR_7013, SR_7020, SR_7030, SR_7040 and SR_8000.

Function The ClientManager has several functions:

- Receive the commands from the ClientTranslator or the IOProcessor.
- Perform the desired actions.
- Update/Read the database.
- Send the requested results back to the ClientTranslator or the IOProcessor.

Subordinates This component consists of a set of classes.

Dependencies None.

Interfaces

1. IDCCommandInput (see (B1) in figure 4.1)

    - `Shutdown()`
      Signals the ClientManager that the dispatcher will shut down.

2. IDUserCommands (see (B2) in figure 4.1)

    - `LogIn(String username, String password)`
      This method should be called when a login request is received. The login will be validated.

    - `RegisterRequest(String username, String password)`
      This method should be called when a user wants to register himself to the SPINGRID system.

3. IDProjectAdminCommands (see (B2) in figure 4.1)

    - `ApproveJobProvider(String projectname, String jobprovider)`
      This method should be called to approve a job provider for the project.

- `DisapproveJobProvider(String projectname, String jobprovider)`
  This method should be called to disapprove a job provider for the project.

- `ApproveResourceProvider(String projectname, String resourceprovider)`
  This method should be called to approve a resource provider for the project.

- `DisapproveResourceProvider(String projectname, String resourceprovider)`
  This method should be called to disapprove a resource provider for the project.

- `ApproveDataSet(String projectname, String dataset)`
  This method should be called to approve a data set for the project.

- `DisapproveDataSet(String projectname, String dataset)`
  This method should be called to disapprove a data set fom the project.

- `ApproveDataProvider(String projectname, String dataprovider)`
  This method should be called to approve a data provider for the project.

- `DisapproveDataProvider(String projectname, String dataprovider)`
  This method should be called to disapprove a data provider for the project.

- `ApproveApplication(String projectname, String application)`
  This method should be called to approve an application for the project.

- `DisapproveApplication(String projectname, String application)`
  This method should be called to disapprove an application for the project.

- `ApproveApplicationProvider(String projectname, String applicationprovider)`
  This method should be called to approve an application provider for the project.

- `DisapproveApplicationProvider(String projectname, String applicationprovider)`
  This method should be called to disapprove an application provider for the project.

- `AllowOwnApplication(String projectname, String jobprovider)`
  This method should be called to allow a job provider to use his private applications for the project.

- `DisallowOwnApplication(String projectname, String jobprovider)`
  This method should be called to disallow a job provider to use his private applications for the project.

- `RemoveJob(int job)`
  This method should be called to remove a job from the SPINGRID system.

- `JobList GetJobList(String projectname)`
  This method should be called to get a list of jobs from the given project.
  Results:
  - A list with all the jobs in the given project.

4. DISystemAdminCommands (see (B2) in figure 4.1)

- `AddApplicationProvider(String username)`
  This method should be called to add an application provider to the SPINGRID system.

- `RemoveApplicationProvider(String username)`
  This method should be called to remove an application provider from the SPINGRID system.

- `AddDataProvider(String username)`
  This method should be called to add a data provider to the SPINGRID system.

- `RemoveDataProvider(String username)`
  This method should be called to remove a data provider from the SPINGRID system.

- `AddProjectAdmin(String username, String projectname)`
  This method should be called to add a project admin to the SPINGRID system.

- `RemoveProjectAdmin(String username, String projectname)`
  This method should be called to remove a project admin from the SPINGRID system.

- `AddProject(String projectname, String username)`
  This method should be called to add a project to the SPINGRID system. The given user is made project admin of the project.

- `RemoveProject(String projectname)`
  This method should be called to remove a project from the SPINGRID system. If there are job providers who where only allowed to provide jobs for the given project then they will loose the role of job provider. If the project admin was only project admin of the given project then he will loose the role of project admin.

- `JobList GetJobList()`
  This method should be called to get a list of jobs from the SPINGRID system.
  Results:
  - A list with all the jobs in the SPINGRID system.

- `UserList GetProjectAdmins()`
  This method should be called to get a list of project admins from the SPINGRID system.
  Results:
  - A list with all the project admins from the SPINGRID system.

- `UserList GetApprovedApplicationProviders(String projectname)`
  This method should be called to get a list of approved application providers from the given project.
  Results:
  - A list with all the application providers in the given project.

- `UserList GetApprovedDataProviders(String projectname)`
  This method should be called to get a list of approved data providers from the given project.
  Results:
  - A list with all the data providers in the given project.

- `UserList GetResourceCalculations(String projectname)`
  This method should be called to get a list of resources which have worked in the given project.
  Results:
  - A list with the resources which have worked in the given project.

5. IDJobProviderCommands (see (B2) in figure 4.1)

- `OfferJob(String projectname, JobDefinition job)`
  This method should be called to offer a job for the given project.

- `RemoveJob(int job)`
  This method should be called to remove a job from the SPINGRID system.

- `URL GetResults(int job)`
  This method should be called to get the results of the given job.
  Results:
  - This will be an URL to the location where the result can be found.

- `JobList GetJobList()`
  This method should be called to get a list of jobs from the given job provider.
  Results:
  - A list with the jobs which are provided by the given job provider.

- `ApplicationList GetApplicationList()`
  This method should be called to get a list of applications which the job provider can use.
  Results:
  - A list with the applications which the job provider can use.

6. IDApplicationProviderCommands

   - `AddApplication(URL application)`
     This method should be called to add an application to the SPINGRID system.

   - `RemoveApplication(String application)`
     This method should be called to remove an application from the SPINGRID system.

   - `ApproveApplicationToProject(String application, String projectname)`
     This method should be called to approve an application for the given project.

   - `DisapproveApplicationToProject(String application, String projectname)`
     This method should be called to disapprove an application for the given project.

   - `ApproveJobProvider(String jobprovider)`
     This method should be called to approve a job provider to use his applications.

   - `DisapproveJobProvider(String jobprovider)`
     This method should be called to disapprove a job provider to use his applications.

   - `ApplicationList GetApplicationList()`
     This method should be called to get a list of all the application provider's provided applications.
     Results:
     - A list with the application provider's provided applications.

   - `ProjectList GetProjectList(String applicationname)`
     This method should be called to get a list of projects which use the given application.
     Results:
     - A list with the projects using the application.

   - `int GetTotalUsed(String application, String projectname)`
     This command should be called to see how often one of the application provider's application is used by a project.
     Results:
     - A integer that has the value of how often the application is used by a project.

7. IDDataProviderCommands (see (B2) in figure 4.1)

   - `AddData(URL data)`
     This method should be called to add data to the SPINGRID system.

   - `RemoveData(String data)`
     This method should be called to remove data from the SPINGRID system.

- `ApproveDataToProject(String data, String projectname)`
  This method should be called to approve data for the given project.

- `DisapproveDataToProject(String data, String projectname)`
  This method should be called to disapprove data for the given project.

- `ApproveJobProvider(String jobprovider)`
  This method should be called to approve a job provider to use the data of the data provider.

- `DisapproveJobProvider(String jobprovider)`
  This method should be called to disapprove a job provider to use the data of the data provider.

- `DataList GetDataList()`
  This method should be called to get a list of all the data provider's provided data.
  Results:
  - A list with the data provider's provided data.

- `ProjectList GetProjectList()`
  This method should be called to get a list of projects which use the data of the data provider.
  Results:
  - A list with the projects using the data provider's data.

- `ProjectList GetApplicationList()`
  This method should be called to get a list of applications which use the data of the data provider.
  Results:
  - A list with the applications using the data provider's data.

5.2.5 AgentCommunicator

Type The AgentCommunicator is a component in the dispatcher application of the SPINGRID project.

Purpose AgentCommunicator implements software requirements SR_3010, SR_3020, SR_3021, SR_3040, SR_3050, SR_3051, SR_3060, SR_3061, SR_3070 and SR_3071.

Function This component has the function to communicate with the agent. This is one of the two components in the dispatcher application able to communicate with the outside world.

Subordinates This component consists of a set of classes:

1. Receiver
   Contains a loop that receives HTTP requests from an agent and creates instances of a communication class.

2. Communication
   Handles the request, uses a blocking method to get the result from the dispatcher and sends this back to the agent.

Dependencies HTTP Message component.

Interfaces None.

5.2.6 AgentTranslator

Type The AgentTranslator is a component in the dispatcher application of the SPINGRID project.

Purpose AgentTranslator implements software requirements SR_3010, SR_3020, SR_3021, SR_3040, SR_3050, SR_3051, SR_3060, SR_3061, SR_3070 and SR_3071.

Function The ClientTranslator has several functions:

- Receive GRP's from the AgentCommunicator.
- Send commands to the AgentManager.
- Translate commands to generic packages and generic packages to commands.

Subordinates This component consists of a set of classes.

Dependencies None.

Interface

1. IDAPackageInput (see (F1) in figure 4.1)
   - `GRP PerformCommand(GSP Package)`
     Returns the result package of a command, after the package is translated into a command and it is performed.

5.2.7 AgentManager

The AgentManager receives messages and commands from AgentTranslator. The messages can be either a request for a job, a 'working-update' and a 'job-completed' message. The commands are used to perform the desired actions. The AgentManager contacts the database to retrieve and update information about jobs and change settings.

The dispatcher needs to know which agent changes the settings and calculates a job. This means that an agent needs some kind of agent identification and that this identification needs to be provided to the methods listed below as an argument. These arguments are left out of this document to keep the methods less complex. The identification will be further explained in the the [DDD].

Type The AgentManager is a component in the dispatcher application of the SPINGRID project.

Purpose AgentManager implements software requirements SR_3010, SR_3020, SR_3021, SR_3040, SR_3050, SR_3051, SR_3060, SR_3061, SR_3070 and SR_3071.

Function The AgentManager has several functions:

- Receive the commands from the AgentTranslator or the IOProcessor
- Perform the desired actions
- Update/Read the database
- Send the requested results back to the AgentTranslator or the IOProcessor
- Manages the agent/job status

Subordinates This component consists of a set of classes.

Dependencies None.

Interface

1. IDACommandInput (see (C1) in figure 4.1)

    - `Shutdown()`
      Signals the AgentManager that the dispatcher will shut down.

1. IDResourceProviderCommands (see (C2) in figure 4.1)

- `AddProject(String project)`
  This method should be called to allow jobs from the given project to be calculated by the resource.

- `RemoveProject(String project)`
  This method should be called to disallow jobs from the given project to be calculated by the resource.

- `ApproveApplication(String application)`
  This method should to approve an application to be executed by the resource.

- `DisapproveApplication(String application)`
  This method should be called to disapprove an application to executed by the resource.

- `ApproveApplicationProvider(String applicationprovider)`
  This method should be called to approve the applications of an application provider to be executed by the resource.

- `String DisapproveApplicationProvider(String applicationprovider)`
  This method should be called to disapprove the applications of an application provider to be executed by the resource.

- `ApproveJobProvider(String jobprovider)`
  This method should be called to approve the jobs of a job provider to be calculated by the resource.

- `String DisapproveJobProvider(String jobprovider)`
  This method should be called to disapprove the jobs of a job provider to be calculated by the resource.

- `ProjectList GetUsing()`
  This method should be called to get a list of project which use the resource.
  Results:
  - A list with the projects using the resource provider's resource.

- `PollResponse Poll(PollRequest polling)`
  This method should be called when a resource polls the dispatcher for work, tells the dispatcher that he is busy or tells the dispatcher that the job has been completed.
  Results:
  - According to the status, this will be an URL of a job or a message.

5.2.8 DatabaseManager

Type The DatabaseManager is a component in the dispatcher application of the SPINGRID project.

Purpose DatabaseManager implements software requirements SR_2010, SR_2011, SR_2020, SR_2021, SR_2030, SR_2031, SR_2040, SR_2041, SR_2060, SR_2070, SR_2080, SR_2090, SR_2100, SR_3010, SR_3020, SR_3021, SR_3040, SR_3050, SR_3051, SR_3060, SR_3061, SR_3070, SR_3071, SR_4010, SR_4011, SR_4020, SR_4021, SR_4030, SR_4031, SR_4032, SR_4033, SR_4040, SR_4041, SR_4042, SR_4043, SR_4050, SR_4051, SR_4060, SR_4070, SR_5010, SR_5011, SR_5020, SR_5030, SR_5040, SR_5050, SR_5021, SR_5070, SR_5071, SR_6010, SR_6011, SR_6020, SR_6021, SR_6030, SR_6040, SR_6050, SR_6060, SR_6061, SR_7010, SR_7011, SR_7012, SR_7013, SR_7020, SR_7030, SR_7040 and SR_8000.

Function The DatabaseManager has several functions:

- Responsible of all the communication with the SQL-database.
- Perform update/read requests of the ClientManager and the AgentManager.

Subordinates This component consist of a set of classes.

Dependencies SQL-database.

Interfaces

1. IDQueryHandler (see (D1) in figure 4.1)
   - `QueryResult PerformQuery(String query)`
     This method should be called when a query needs be sent to the database.
     Results:
     - The result of the query.

5.3 Client Components

5.3.1 IOProcessor

Type The IOProcessor is a component in the client application of the SPINGRID project.

Purpose IOProcessor implements software requirements SR_2010, SR_2011, SR_2020, SR_2021, SR_2030, SR_2031, SR_2040, SR_2041, SR_2060, SR_2070, SR_2080, SR_2090, SR_2100, SR_4010, SR_4011, SR_4020, SR_4021, SR_4030, SR_4031, SR_4032, SR_4033, SR_4040, SR_4041, SR_4042, SR_4043, SR_4050, SR_4051, SR_4060, SR_4070, SR_5010, SR_5011, SR_5020, SR_5021, SR_5030, SR_5040, SR_5050, SR_5070, SR_5071, SR_6010, SR_6011, SR_6020, SR_6021, SR_6030, SR_6040, SR_6050, SR_6060, SR_6061, SR_7010, SR_7011, SR_7012, SR_7013, SR_7020, SR_7030, SR_7040, SR_8000 and SR_9030.

Function This component has the function to process inputs from the outside world and process outputs from the other components of the client.

Subordinates This component consists of a set of classes.

Dependencies None.

Interfaces

1. ICInputProcessor (see (A1) in figure 4.2)

   - `Input(String[] args)`
     This method should be called when the user of the client application has a sequence of characters ready to process. The method calls the appropriated method from the Translator component when the input is valid.

2. ICOutputProcessor (see (A2) in figure 4.2)

   - `Output(String str)`
     This method should be called when the Translator wants to send a message to the user of the client application. The sequence of characters is sent to standard output.

5.3.2 Translator

Type The Translator is a component in the SPINGRID project and is part of the client application.

Purpose Translator implements software requirements SR_2010, SR_2011, SR_2020, SR_2021, SR_2030, SR_2031, SR_2040, SR_2041, SR_2060, SR_2070, SR_2080, SR_2090, SR_2100, SR_4010, SR_4011, SR_4020, SR_4021, SR_4030, SR_4031, SR_4032, SR_4033, SR_4040, SR_4041, SR_4042, SR_4043, SR_4050, SR_4051, SR_4060, SR_4070, SR_5010, SR_5011, SR_5020, SR_5021, SR_5030, SR_5040, SR_5050, SR_5070, SR_5071, SR_6010, SR_6011, SR_6020, SR_6021, SR_6030, SR_6040, SR_6050, SR_6060, SR_6061, SR_7010, SR_7011, SR_7012, SR_7013, SR_7020, SR_7030, SR_7040, and SR_8000.

Function When a method in the Translator is called, it translates the parameters of the method into a GSP. The GSPs are then send to the Communicator. GRPs from the Communicator are evaluated so that the appropriate message can be send to the user

Subordinates This component consists of a set of classes.

Dependencies None.

Interfaces

1. ICUserCommands (see (B1) in figure 4.2)

   - `sendLogIn(String username, String password)`
     This method should be called when a user wants to log in.

   - `sendRegisterRequest(String username, String password)`
     This method should be called when a user wants to register himself to the dispatcher.

2. ICProjectAdminCommands (see (B1) in figure 4.2)

   - `sendApproveJobProvider(String username, String projectname)`
     This method should be called to approve a job provider for the project.

   - `sendDisapproveJobProvider(String username, String projectname)`
     This method should be called to disapprove a job provider for the project.

   - `sendApproveResourceProvider(String projectname, String resourceprovider)`
     This method should be called to approve a resource provider for the project.

   - `sendDisapproveResourceProvider(String projectname, String resourceprovider)`
     This method should be called to disapprove a resource provider for the project.

   - `sendApproveDataSet(String projectname, String dataset)`
     This method should be called to approve a data set for the project.

   - `sendDisapproveDataSet(String projectname, String dataset)`
     This method should be called to disapprove a data set fom the project.

   - `sendApproveDataProvider(String projectname, String dataprovider)`
     This method should be called to approve a data provider for the project.

- `sendDisapproveDataProvider(String projectname, String dataprovider)`
  This method should be called to disapprove a data provider for the project.

- `sendApproveApplication(String projectname, String application)`
  This method should be called to approve an application for the project.

- `sendDisapproveApplication(String projectname, String application)`
  This method should be called to disapprove an application for the project.

- `sendApproveApplicationProvider(String projectname, String applicationprovider)`
  This method should be called to approve an application provider for the project.

- `sendDisapproveApplicationProvider(String projectname, String applicationprovider)`
  This method should be called to disapprove an application provider for the project.

- `sendAllowOwnApplication(String projectname, String jobprovider)`
  This method should be called to allow a job provider to use his private applications for the project.

- `sendDisallowOwnApplication(String projectname, String jobprovider)`
  This method should be called to disallow a job provider to use his private applications for the project.

- `sendRemoveJob(int job)`
  This method should be called to remove a job from the SPINGRID system.

- `JobList sendGetJobList(String projectname)`
  This method should be called to get a list of jobs from the given project.
  Results:
  - A list with all the jobs in the given project.

3. ICSystemAdminCommands (see (B1) in figure 4.2)

- `sendAddApplicationProvider(String username)`
  This method should be called to add an application provider to the SPINGRID system.

- `sendRemoveApplicationProvider(String username)`
  This method should be called to remove an application provider from the SPINGRID system.

- `sendAddDataProvider(String username)`
  This method should be called to add a data provider to the SPINGRID system.

- `sendRemoveDataProvider(String username)`
  This method should be called to remove a data provider from the SPINGRID system.

- `sendAddProjectAdmin(String username, String projectname)`
  This method should be called to add a project admin to the SPINGRID system.

- `sendRemoveProjectAdmin(String username, String projectname)`
  This method should be called to remove a project admin from the SPINGRID system.

- `sendAddProject(String projectname, String username)`
  This method should be called to add a project to the SPINGRID system. The given user is made project admin of the project.

- `sendRemoveProject(String projectname)`
  This method should be called to remove a project from the SPINGRID system. If there are job providers who where only allowed to provide jobs for the given project then they will loose the role of job provider. If the project admin was only project admin of the given project then he will loose the role of project admin.

- `JobList sendGetJobList()`
  This method should be called to get a list of jobs from the SPINGRID system.
  Results:
  - A list with all the jobs in the SPINGRID system.

- `UserList sendGetProjectAdmins()`
  This method should be called to get a list of project admins from the SPINGRID system.
  Results:
  - A list with all the project admins from the SPINGRID system.

- `UserList sendGetApprovedApplicationProviders(String projectname)`
  This method should be called to get a list of approved application providers from the given project.
  Results:
  - A list with all the application providers in the given project.

- `UserList sendGetApprovedDataProviders(String projectname)`
  This method should be called to get a list of approved data providers from the given project.
  Results:
  - A list with all the data providers in the given project.

- `UserList sendGetResourceCalculations(String projectname)`
  This method should be called to get a list of resources which have worked in the given project.
  Results:

  - A list with the resources which have worked in the given project.

4. ICJobProviderCommands (see (B1) in figure 4.2)

   - `sendOfferJob(String projectname, JobDefinition job)`
     This method should be called to offer a job for the given project.

   - `sendRemoveJob(int job)`
     This method should be called to remove a job from the SPINGRID system.

   - `URL sendGetResults(int job)`
     This method should be called to get the results of the given job.
     Results:
     - This will be an URL to the location where the result can be found.

   - `JobList sendGetJobList()`
     This method should be called to get a list of jobs from the given job provider.
     Results:
     - A list with the jobs which are provided by the given job provider.

   - `ApplicationList sendGetApplicationList()`
     This method should be called to get a list of applications which the job provider can use.
     Results:
     - A list with the applications which the job provider can use.

5. ICApplicationProviderCommands (see (B1) in figure 4.2)

   - `sendAddApplication(URL application)`
     This method should be called to add an application to the SPINGRID system.

   - `sendRemoveApplication(String application)`
     This method should be called to remove an application from the SPINGRID system.

   - `sendApproveApplicationToProject(String application, String projectname)`
     This method should be called to approve an application for the given project.

   - `sendDisapproveApplicationToProject(String application, String projectname)`
     This method should be called to disapprove an application for the given project.

   - `sendApproveJobProvider(String jobprovider)`
     This method should be called to approve a job provider's own applications.

   - `sendDisapproveJobProvider(String jobprovider)`
     This method should be called to disapprove a job provider's own applications.

- `ApplicationList sendGetApplicationList()`
  This method should be called to get a list of all the application provider's provided applications.
  Results:
  - A list with the application provider's provided applications.

- `ProjectList sendGetProjectList(String applicationname)`
  This method should be called to get a list of projects which use the given application.
  Results:
  - A list with the projects using the application.

- `int sendGetTotalUsed(String application, String projectname)`
  This command should be called to see how often one of the application provider's application is used by a project.
  Results:
  - A integer that has the value of how often the application is used by a project.

6. ICDataProviderCommands (see (B1) in figure 4.2)

- `AddData(URL data)`
  This method should be called to add data to the SPINGRID system.

- `RemoveData(String data)`
  This method should be called to remove data from the SPINGRID system.

- `ApproveDataToProject(String data, String projectname)`
  This method should be called to approve data for the given project.

- `DisapproveDataToProject(String data, String projectname)`
  This method should be called to disapprove data for the given project.

- `ApproveJobProvider(String jobprovider)`
  This method should be called to approve a job provider to use the data of the data provider.

- `DisapproveJobProvider(String jobprovider)`
  This method should be called to disapprove a job provider to use the data of the data provider.

- `DataList GetDataList()`
  This method should be called to get a list of all the data provider's provided data.
  Results:
  - A list with the data provider's provided data.

- `ProjectList GetProjectList()`
  This method should be called to get a list of projects which use the data of the data provider.
  Results:
  - A list with the projects using the data provider's data.

- `ProjectList GetApplicationList()`
  This method should be called to get a list of applications which use the data of the data provider.
  Results:
  - A list with the applications using the data provider's data.

5.3.3 Communicator

Type The Communicator is a component in the SPINGRID project and is part of all applications.

Purpose Communicator implements software requirements SR_2010, SR_2011, SR_2020, SR_2021, SR_2030, SR_2031, SR_2040, SR_2041, SR_2060, SR_2070, SR_2080, SR_2090, SR_2100, SR_4010, SR_4011, SR_4020, SR_4021, SR_4030, SR_4031, SR_4032, SR_4033, SR_4040, SR_4041, SR_4042, SR_4043, SR_4050, SR_4051, SR_4060, SR_4070, SR_5010, SR_5011, SR_5020, SR_5021, SR_5030, SR_5040, SR_5050, SR_5070, SR_5071, SR_6010, SR_6011, SR_6020, SR_6021, SR_6030, SR_6040, SR_6050, SR_6060, SR_6061, SR_7010, SR_7011, SR_7012, SR_7013, SR_7020, SR_7030, SR_7040, and SR_8000.

Function This component has the function to communicate with the dispatcher.

Subordinates This component consists of a set of classes.

Dependencies HTTP Message component.

Interfaces

1. ICCommunicator (see (C1) in figure 4.2)

   - `GRP Send(GSP package, InetAddress addr)`
     This method should be called when a GSP package (a list of pairs of variables and values) needs to be send to another host. To accomplish this, the method uses converts the GSPs into messages that conform to the protocol and then sends those to the given host. The method then blocks and waits for a response. If a response is not received in time, an exception is thrown. After the response is received the data is converted to a GRP (combination of a string containing the status code and a string containing a message) and returned.
     Results:

– A GRP which contains the response given by the host.

5.4 Agent Components

5.4.1 IOProcessor

Type The IOProcessor is a component in the agent application of the SPINGRID project.

Purpose IOProcessor implements software requirements SR_3010, SR_3020, SR_3021, SR_3030, SR_3031, SR_3040, SR_3050, SR_3051, SR_3060, SR_3061, SR_3070, SR_3071, SR_8000 and SR_9030.

Function This component manages the input received from and output given to the user. It redirects input to and receives output from the Distributor.

Subordinates This component consists of a set of classes.

Dependencies None.

Interfaces

1. IAInputProcessor (see (A1) in figure 4.3)

   - `Input(String[] args)`
     This method should be called when the user of the agent application has a sequence of strings ready to process. The method calls the appropriated method from IADistributorForIOProcessor, which is implemented in the Distributor.

2. IAOutputProcessor (see (A2) in figure 4.3)

   - `Output(String str)`
     This method should be called when the Distributor wants to send a message to the user of the agent application. The sequence of characters is sent to standard output.

5.4.2 Distributor

Type The Distributor is a component in the the agent application of the SPINGRID project.

Purpose Distributor implements software requirements SR_3010, SR_3020, SR_3021, SR_3030, SR_3031, SR_3040, SR_3050, SR_3051, SR_3060, SR_3061, SR_3070 and SR_3071.

Function This component receives tasks from the IOProcessor and the JobScheduler and passes it to the Translator. This component is separated from the Translator for multi-threading reasons.

Subordinates This component consists of a set of classes.

Dependencies None.

Interfaces

1. IADistributorForIOProcessor (see (B1) in figure 4.3)

   - `StartScheduler()`
     This method calls Start() in IAJobScheduler which is implemented in JobScheduler.

   - `StopScheduler()`
     This method calls Stop() in IAJobScheduler which is implemented in JobScheduler.

   - `AddInterval(Interval a, Interval b)`
     This method calls AddInterval(Interval a, Interval b) in IAJobScheduler which is implemented in JobScheduler.

   - `RemoveInterval(Interval a, Interval b)`
     This method calls RemoveInterval(Interval a, Interval b) in IAJobScheduler which is implemented in JobScheduler.

   - `sendAddProject(String projectname)`
     This method calls sendAddProject(String projectname) in IATranslator which is implemented in Translator.

   - `sendRemoveProject(String projectname)`
     This method calls sendRemoveProject(String projectname) in IATranslator which is implemented in Translator.

   - `sendApproveApplication(String applicationname)`
     This method calls sendApproveApplication(String applicationname) in IATranslator which is implemented in Translator.

   - `sendDisapproveApplication(String applicationname)`
     This method calls sendDisapproveApplication(String applicationname) in IATranslator which is implemented in Translator.

- `sendApproveApplicationProvider(String applicationprovidername)`
  This method calls sendApproveApplicationProvider(String applicationprovidername) in IATranslator which is implemented in Translator.

- `sendDisapproveApplicationProvider(String applicationprovidername)`
  This method calls sendDisapproveApplicationProvider(String applicationprovidername) in IATranslator which is implemented in Translator.

- `sendApproveJobProvider(String jobprovidername)`
  This method calls sendApproveJobProvider(String jobprovidername) in IATranslator which is implemented in Translator.

- `sendDisapproveJobProvider(String jobprovidername)`
  This method calls sendDisapproveJobProvider(String jobprovidername) in IATranslator which is implemented in Translator.

- `ProjectList sendGetUsing()`
  This method calls sendGetUsing() in IATranslator which is implemented in Translator.

2. IADistributorForJobScheduler (see (B2) in figure 4.3)

   - `sendPoll()`
     This method calls sendPoll() in IATranslator which is implemented in Translator.

5.4.3 JobScheduler

Type The JobScheduler is a component in the agent application of the SPINGRID project.

Purpose JobScheduler implements software requirements SR_3010, SR_3030 and SR_3031.

Function This component is responsible for scheduling jobs. This means that the component needs to:

- poll for jobs.
- download data through the DataManager which is needed to execute jobs.
- execute jobs using the JobExecutor.
- upload data through the DataManager when a job is finished executing.
- signal the dispatcher when a job is finished executing.

Subordinates This component consists of a set of classes.

Dependencies None.

Interfaces

1. IAJobSchedulerForDistributor (see (C1) in figure 4.3)
   - `Start()`
     This method is called when the JobScheduler should begin with scheduling jobs.
   - `Stop()`
     This method is called when the JobScheduler should stop with scheduling jobs.
   - `AddInterval(Interval a, Interval b)`
     This method is called when an interval should be added when the resource can be used.
   - `RemoveInterval(Interval a, Interval b)`
     This method is called when an interval should be removed when the resource can be used.
   - `receivePoll(Object Job)`
     This method is called when the answer to a poll is received.
2. IAJobSchedulerForJobExecutor (see (C2) in figure 4.3)
   - `GiveResult(String Status)`
     This method should be called when the status of a job has changed.
3. IAJobSchedulerForDataManager (see (C3) in figure 4.3)
   - `GiveDownloadResult(String Status)`
     This method should be called when the status of a download order has changed.
   - `GiveUploadResult(String Status)`
     This method should be called when the status of a upload order has changed.

5.4.4 JobExecutor

Type The JobExecutor is a component in the agent application of the SPINGRID project.

Purpose JobExecutor implements no software requirements.

Function This component executes jobs. A signal will be send to the JobScheduler when a job is done executing.

Subordinates This component consists of a set of classes.

Dependencies None.

Interfaces

1. IAJobExecutor (see (E1) in figure 4.3)

   - `StartJob(int job)`
     This method should be called when a job needs to start executing.

   - `StopJob(int job)`
     This method should be called when a job needs to stop executing.

5.4.5 DataManager

Type The DataManager is a component in the SPINGRID project and is part of the agent application.

Purpose DataManager implements no software requirements.

Function This component will download data needed to perform jobs and will upload results when jobs are completed.

Subordinates This component consists of a set of classes.

Dependencies None.

Interfaces

1. IADataManager (see (D1) in figure 4.3)

   - `Download(String frompath, String topath)`
     This method should be called when data located in frompath should be downloaded to topath.

- `Upload(String frompath, String topath)`
  This method should be called when data located in frompath should be uploaded to topath.

5.4.6 Translator

Type The Translator is a component in the agent application of the SPINGRID project.

Purpose Translator implements software requirements SR_3010, SR_3020, SR_3021, SR_3040, SR_3050, SR_3051, SR_3060, SR_3061, SR_3070 and SR_3071.

Function When a method in the Translator is called, it translates the parameters of the method into a GSP. The GSPs are then send to the Communicator. GRPs from the Communicator are evaluated so that the appropriate message can be send to the user through the Distributor.

Subordinates This component consists of a set of classes.

Dependencies None.

Interfaces

1. IAResourceProviderCommands (see (F1) in figure 4.3)

   - `sendAddProject(String project)`
     This method should be called to allow jobs from the given project to be calculated by the resource.

   - `sendRemoveProject(String project)`
     This method should be called to disallow jobs from the given project to be calculated by the resource.

   - `sendApproveApplication(String application)`
     This method should to approve an application to be executed by the resource.

   - `sendDisapproveApplication(String application)`
     This method should be called to disapprove an application to executed by the resource.

   - `sendApproveApplicationProvider(String applicationprovider)`
     This method should be called to approve the applications of an application provider

to be executed by the resource.

- `sendDisapproveApplicationProvider(String applicationprovider)`
  This method should be called to disapprove the applications of an application provider to be executed by the resource.

- `sendApproveJobProvider(String jobprovider)`
  This method should be called to approve the jobs of a job provider to be calculated by the resource.

- `sendDisapproveJobProvider(String jobprovider)`
  This method should be called to disapprove the jobs of a job provider to be calculated by the resource.

- `ProjectList sendGetUsing()`
  This method should be called to get a list of project which use the resource.
  Results:
  - A list with the projects using the resource provider's resource.

- `PollResponse sendPoll(PollRequest polling)`
  This method should be called when a resource polls the dispatcher for work, tells the dispatcher that he is busy or tells the dispatcher that the job has been completed.
  Results:
  - According to the status, this will be an URL of a job or a message.

5.4.7 Communicator

Type The Communicator is a component in the agent application of the SPINGRID project.

Purpose Communicator implements software requirements SR_3010, SR_3020, SR_3021, SR_3040, SR_3050, SR_3051, SR_3060, SR_3061, SR_3070 and SR_3071.

Function This component has the function to communicate with the dispatcher.

Subordinates This component consists of a set of classes.

Dependencies HTTP Message component.

Interfaces

1. IACommunicator (see (G1) in figure 4.2)

- `GRP Send(GSP package, InetAddress addr)`
  This method should be called when a GSP package (a list of pairs of variables and values) needs to be send to another host. To accomplish this, the method uses converts the GSPs into messages that conform to the protocol and then sends those to the given host. The method then blocks and waits for a response. If a response is not received in time, an exception is thrown. After the response is received the data is converted to a GRP (combination of a string containing the status code and a string containing a message) and returned.
  Results:
  - A GRP which contains the response given by the host.

Chapter 6

Feasibility and resource estimates

6.1 Resources to build the system

The following resources are used to build the various SPINGRID software applications:

- 7 NEC Versa P520 notebooks with an 1.4GHz Intel Celeron processor, 512 MB primary and 40 GB seondary storage. These systems are mainly used for writing documents, coding and testing of the system.
- 1 Dell computer with an Intel Pentium IV 3,0 GHz, 1024 MB primary and 40 GB secondary storage, running Windows XP Pro. This system is mainly used for writing documents, coding and testing of the system.
- 1 Dell computer with an Intel Pentium IV 3,0 GHz, 1024 MB primary and 40 GB secondary storage, running Debian GNU/Linux 3.1. This system is mainly used for writing documents, coding and testing of the system.

6.2 Resources to operate the system

The software applications will be used on multiple computer systems, but all these systems will have at least the following resources (depending which application they will run). The client application does not have any requirements besides the general requirements.

6.2.1 General

- Windows XP, Mac OS X or Linux 2.4
- Sun JRE 1.4.2 or 1.5

6.2.2 Dispatcher

- Intel Pentium IV 2 GHz or equivalent processor

- 2 GB RAM
- 2 MBit/s network connection
- Dedicated machine
- Linux

6.2.3 Agent

- Intel Pentium II 300 MHz or equivalent processor, G4 700 MHz or equivalent
- 128 MB RAM
- 256 available hard disk space

6.3 Resources to maintain the system

Although the development team will not perform any maintenance on the SPINGRID software after the transfer phase, in the case that maintenance is done, it is strongly recommended to use at least the configuration used during the development phase.

Chapter 7

Requirements traceability matrix

There are a couple of software requirements (described in [SRD]) which are not coupled with one or more methods in this document. This happened because of the nature of those software requirements. Most of the time this is the case when the whole SPINGRID system implements the software requirement. But there are also software requirements for which it was not clear how to implement them. The software requirements which are not coupled with methods in this document are described below.

- **[SR_2050] The system admin has the option to change the system settings.**
  It is unclear at this time what the different system settings are.

- **[SR_5060] An application has an attribute called *Characteristics* which contains the characteristics that the application requires.**
  It is unclear how this will be implemented.

- **[SR_7110]: A private application is provided by precisely one job provider.**
  This is a restriction on the database and therefor cannot be coupled with a method.

- **[SR_7120]: A private data object is provided by precisely one job provider.**
  This is a restriction on the database and therefor cannot be coupled with a method.

- **[SR_7130]: A project has at least one project admin.**
  This is a restriction on the database and therefor cannot be coupled with a method. Note that in this case it is possible to implement the software requirement with methods. Then the methods which handle the action of removing a project admin from a project and the action that adds a project to the SPINGRID system needs to implement the software requirement.

- **[SR_7150]: The SPINGRID shall have one system admin.**
  This is a restriction on the database and therefor cannot be coupled with a method.

- **[SR_9000]: The SPINGRID system shall implement a computational grid.**
  The whole SPINGRID system is responsible for implementing this software requirement.

- **[SR_9010]: The system requirements are as described in section 2.4.1.**
  The whole SPINGRID system is responsible for implementing this software requirement.

- **[SR_9020]: The language used in the product will be English.**
  The whole SPINGRID system is responsible for implementing this software requirement.
- **[SR_9031]: Interaction with the system will be provided by a web-based user interface.**
  There are no methods coupled with this software requirement because this software requirement has a low priority. The input/output-processor components in the three applications can implement these when desired.
- **[SR_9040]: The SPINGRID system will be able to process at least 40 executing jobs at a time.**
  The whole SPINGRID system is responsible for implementing this software requirement.
- **[SR_9050]: The SPINGRID system will select the resource that will be used for processing a job.**
  The dispatcher application is responsible for implementing this software requirement.
- **[SR_9060]: The SPINGRID system shall only send jobs to resources if their characteristics at least match the characteristics required by the job and application used in the job.**
  The dispatcher application is responsible for implementing this software requirement and that can be achieved by using JSDL.
- **[SR_9070]: The SPINGRID system shall provide a trust model as described in appendix A of [URD].**
  It is unclear if the whole trust model is implemented. Note that the approve and disapprove methods implement this software requirement partly.
- **[SR_9080]: The (un-)installation of the SPINGRID system will not require a computer expert.**
  The whole SPINGRID system is responsible for implementing this software requirement.
- **[SR_9090]: When there are at least 2 dispatchers in the system and one of them disappears, the system will continue without malfunction.**
  The whole SPINGRID system is responsible for implementing this software requirement.
- **[SR_9100]: When all the dispatchers in the system are down and one of them is restarted, the system will continue without malfunction.**
  The whole SPINGRID system is responsible for implementing this software requirement.
- **[SR_9110]: If one of the resources disappears while performing a job, the system will requeue the job.**
  The dispatcher application is responsible for implementing this software requirement.
- **[SR_9120]: A job will be declared failed after it has been requeued for a configurable number of times.**
  The dispatcher application is responsible for implementing this software requirement.
- **[SR_9140]: The SPINGRID system will be implemented in Java according to (a tailored version of) the BSSC Java Coding Standards.**
  The whole SPINGRID system is responsible for implementing this software requirement.

- **[SR_9150]: The system will be able to run for at least a week without interruption.**
  The whole SPINGRID system is responsible for implementing this software requirement.
- **[SR_9170]: The total time in which none of the dispatchers responds will not exceed one hour a day.**
  The whole SPINGRID system is responsible for implementing this software requirement.
- **[SR_9180]: The SPINGRID system is able to sent a notification to the job provider when a job is completed, failed or removed.**
  The whole SPINGRID system is responsible for implementing this software requirement.

All other software requirements, which are not mentioned above, are coupled with one or more methods. This table can be found below this text. A method will be referred by first a character G, C, D or A which stands for a general, client, dispatcher or agent component. Then the component name after which a dot follows and the method name. A * as method name stands for all the methods in that component. When two or more methods have the same name in the same component then a tag is added: (A), (D), (P) or (S) which stands for a method which can be called by the application provider, data provider, project admin, resource provider or the system admin.

7.1 Software Requirements to Architectural Design

Software Requirements	Architectural Design
SR_1210	GJSDLDescription.GetJobName
SR_1220	GJSDLDescription.GetJobDescription
SR_1230	GJSDLDescription.GetGetJobAnnotations
SR_1240	GJSDLDescription.GetGetJobProjects
SR_1310	GJSDLDescription.GetApplicationName
SR_1320	GJSDLDescription.GetApplicationVersion
SR_1330	GJSDLDescription.GetApplicationDescription
SR_1410	GJSDLDescription.GetCandidateHosts
SR_1411	GJSDLDescription.GetFileSystems
SR_1412	GJSDLDescription.GetExclusiveExecution
SR_1413	GJSDLDescription.GetOperatingSystem
SR_1414	GJSDLDescription.GetCPUArchitecture
SR_1415	GJSDLDescription.GetICPUSpeed
SR_1416	GJSDLDescription.GetICPUTime
SR_1417	GJSDLDescription.GetICPUCount
SR_1418	GJSDLDescription.GetINetworkBandwidth
SR_1419	GJSDLDescription.GetIPhysicalMemory
SR_1420	GJSDLDescription.GetIVirtualMemory
SR_1421	GJSDLDescription.GetIDiskSpace
SR_1422	GJSDLDescription.GetTCPUTime
SR_1423	GJSDLDescription.GetTCPUCount

SR_1424	GJSDLDescription.GetTPhysicalMemory
SR_1425	GJSDLDescription.GetTVirtualMemory
SR_1426	GJSDLDescription.GetTDiskSpace
SR_1427	GJSDLDescription.GetTResourceCount
SR_1510	GJSDLDescription.GetDataStaging
SR_1520	GJSDLDescription.GetDataStaging
SR_1530	GJSDLDescription.GetDataStaging
SR_1540	GJSDLDescription.GetDataStaging
SR_1550	GJSDLDescription.GetDataStaging
SR_1560	GJSDLDescription.GetDataStaging
SR_2010	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendAddApplicationProvider, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.AddApplicationProvider, DDatabaseManager.PerformQuery
SR_2011	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendRemoveApplicationProvider, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.RemoveApplicationProvider, DDatabaseManager.PerformQuery
SR_2020	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendAddDataProvider, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.AddDataProvider, DDatabaseManager.PerformQuery
SR_2021	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendRemoveDataProvider, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.RemoveDataProvider, DDatabaseManager.PerformQuery

SR_2030	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendAddProjectAdmin, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.AddProjectAdmin, DDatabaseManager.PerformQuery
SR_2031	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendRemoveProjectAdmin, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.RemoveProjectAdmin, DDatabaseManager.PerformQuery
SR_2040	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendAddProject, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.AddProject, DDatabaseManager.PerformQuery
SR_2041	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendRemoveProject, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.RemoveProject, DDatabaseManager.PerformQuery
SR_2050	See text above.
SR_2060	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendGetJobList(S), CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.GetJobList(S), DDatabaseManager.PerformQuery

SR_2070	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendGetProjectAdmins, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.GetProjectAdmins, DDatabaseManager.PerformQuery
SR_2080	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendGetApprovedApplicationProviders, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.GetApprovedApplicationProviders, DDatabaseManager.PerformQuery
SR_2090	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendGetApprovedDataProviders, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.GetApprovedDataProviders, DDatabaseManager.PerformQuery
SR_2100	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendGetResourceCalculations, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.GetResourceCalculations, DDatabaseManager.PerformQuery
SR_3010	GHTTPMessage.*, AIOProcessor.Input, ADistributor.sendPoll, ATranslator.sendPoll, AJobScheduler.receivePoll, ACommunicator.send, DAgentCommunicator.*, DAgentTranslator.*, DAgentManager.Poll, DDatabaseManager.PerformQuery
SR_3011	GHTTPMessage.*

SR_3020	GHTTPMessage.*, AIOProcessor.Input, ADistributor.sendAddProject, ATranslator.sendAddProject, ACommunicator.send, DAgentCommunicator.*, DAgentTranslator.*, DAgentManager.AddProject, DDatabaseManager.PerformQuery
SR_3021	GHTTPMessage.*, AIOProcessor.Input, ADistributor.sendRemoveProject, ATranslator.sendRemoveProject, ACommunicator.send, DAgentCommunicator.*, DAgentTranslator.*, DAgentManager.RemoveProject, DDatabaseManager.PerformQuery
SR_3030	GHTTPMessage.*, AIOProcessor.Input, ADistributor.AddInterval, AJobScheduler.AddInterval
SR_3031	GHTTPMessage.*, AIOProcessor.Input, ADistributor.RemoveInterval, AJobScheduler.RemoveInterval
SR_3040	GHTTPMessage.*, AIOProcessor.Input, ADistributor.sendGetUsing, ATranslator.sendGetUsing, ACommunicator.send, DAgentCommunicator.*, DAgentTranslator.*, DAgentManager.GetUsing, DDatabaseManager.PerformQuery
SR_3050	GHTTPMessage.*, AIOProcessor.Input, ADistributor.sendApproveApplication, ATranslator.sendApproveApplication, ACommunicator.send, DAgentCommunicator.*, DAgentTranslator.*, DAgentManager.ApproveApplication, DDatabaseManager.PerformQuery

SR_3051	GHTTPMessage.*, AIOProcessor.Input, ADistributor.sendDisapproveApplication, ATranslator.sendDisapproveApplication, ACommunicator.send, DAgentCommunicator.*, DAgentTranslator.*, DAgentManager.DisapproveApplication, DDatabaseManager.PerformQuery
SR_3060	GHTTPMessage.*, AIOProcessor.Input, ADistributor.sendApproveApplicationProvider, ATranslator.sendApproveApplicationProvider, ACommunicator.send, DAgentCommunicator.*, DAgentTranslator.*, DAgentManager.ApproveApplicationProvider, DDatabaseManager.PerformQuery
SR_3061	GHTTPMessage.*, AIOProcessor.Input, ADistributor.sendDisapproveApplicationProvider, ATranslator.sendDisapproveApplicationProvider, ACommunicator.send, DAgentCommunicator.*, DAgentTranslator.*, DAgentManager.DisapproveApplicationProvider, DDatabaseManager.PerformQuery
SR_3070	GHTTPMessage.*, AIOProcessor.Input, ADistributor.sendApproveJobProvider, ATranslator.sendApproveJobProvider, ACommunicator.send, DAgentCommunicator.*, DAgentTranslator.*, DAgentManager.ApproveJobProvider, DDatabaseManager.PerformQuery
SR_3071	GHTTPMessage.*, AIOProcessor.Input, ADistributor.sendDisapproveJobProvider, ATranslator.sendDisapproveJobProvider, ACommunicator.send, DAgentCommunicator.*, DAgentTranslator.*, DAgentManager.DisapproveJobProvider, DDatabaseManager.PerformQuery

SR_4010	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendApproveJobProvider(P), CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.ApproveJobProvider(P), DDatabaseManager.PerformQuery
SR_4011	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendDisapproveJobProvider(P), CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.DisapproveJobProvider(P), DDatabaseManager.PerformQuery
SR_4020	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendApproveResourceProvider, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.ApproveResourceProvider, DDatabaseManager.PerformQuery
SR_4021	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendDisapproveResourceProvider, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.DisapproveResourceProvider, DDatabaseManager.PerformQuery
SR_4030	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendApproveDataset, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.ApproveDataSet, DDatabaseManager.PerformQuery

SR_4031	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendDisapproveDataset, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.DisapproveDataSet, DDatabaseManager.PerformQuery
SR_4032	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendApproveDataProvider, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.ApproveDataProvider, DDatabaseManager.PerformQuery
SR_4033	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendDisapproveDataProvider, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.DisapproveDataProvider, DDatabaseManager.PerformQuery
SR_4040	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendApproveApplication, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.ApproveApplication, DDatabaseManager.PerformQuery
SR_4041	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendDisapproveApplication, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.DisapproveApplication, DDatabaseManager.PerformQuery

SR_4042	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendApproveApplicationProvider, CCommunicator.Send, DClientCommunicator.*, DClientManager.DClientTranslator.*, DClientManager.ApproveApplicationProvider, DDatabaseManager.PerformQuery
SR_4043	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendDisapproveApplicationProvider, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.DisapproveApplicationProvider, DDatabaseManager.PerformQuery
SR_4050	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendAllowOwnApplication, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.AllowOwnApplication, DDatabaseManager.PerformQuery
SR_4051	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendDisallowOwnApplication, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.DisallowOwnApplication, DDatabaseManager.PerformQuery
SR_4060	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendRemoveJob(P), CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.RemoveJob(P), DDatabaseManager.PerformQuery

SR_4070	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendGetJobList(P), CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.GetJobList(P), DDatabaseManager.PerformQuery
SR_5010	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendAddApplication, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.AddApplication, DDatabaseManager.PerformQuery
SR_5011	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendRemoveApplication, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.RemoveApplication, DDatabaseManager.PerformQuery
SR_5020	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendApproveApplicationToProject, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.ApproveApplicationToProject, DDatabaseManager.PerformQuery
SR_5021	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendDisapproveApplicationFromProject, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.DisapproveApplicationToProject, DDatabaseManager.PerformQuery

SR_5030	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendGetApplicationsList(A), CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.GetApplicationList(A), DDatabaseManager.PerformQuery
SR_5040	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendGetTotalUsed, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.GetTotalUsed, DDatabaseManager.PerformQuery
SR_5050	GHTTPMessage.*, CIOProcessor.Input, CTranslator.GetProjectList, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.GetProjectList(A), DDatabaseManager.PerformQuery
SR_5060	See text above.
SR_5070	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendApproveJobProvider(A), CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.ApproveJobProvider(A), DDatabaseManager.PerformQuery
SR_5071	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendDisapproveJobProvider(A), CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.DisapproveJobProvider(A), DDatabaseManager.PerformQuery

SR_6010	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendAddData, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.AddData, DDatabaseManager.PerformQuery
SR_6011	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendRemoveData, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.RemoveData, DDatabaseManager.PerformQuery
SR_6020	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendApproveDataToProject, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.ApproveDataToProject, DDatabaseManager.PerformQuery
SR_6021	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendDisapproveDataFromProject, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.DisapproveDataToProject, DDatabaseManager.PerformQuery
SR_6030	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendGetProjectList, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.GetProjectList(D), DDatabaseManager.PerformQuery

SR_6040	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendGetDataList, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.GetDataList, DDatabaseManager.PerformQuery
SR_6050	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendGetApplicationsList(D), CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.GetApplicationList(D), DDatabaseManager.PerformQuery
SR_6060	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendApproveJobProvider(D), CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.ApproveJobProvider(D), DDatabaseManager.PerformQuery
SR_6061	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendDisapproveJobProvider(D), CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.DisapproveJobProvider(D), DDatabaseManager.PerformQuery
SR_7010	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendOfferJob, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.OfferJob, DDatabaseManager.PerformQuery

SR_7011	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendRemoveJob(J), CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.RemoveJob(J), DDatabaseManager.PerformQuery
SR_7012	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendOfferJob, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.OfferJob, DDatabaseManager.PerformQuery
SR_7013	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendOfferJob, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.OfferJob, DDatabaseManager.PerformQuery
SR_7020	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendGetApplicationList, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.GetApplicationList(J), DDatabaseManager.PerformQuery
SR_7030	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendGetResults, CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.GetResults, DDatabaseManager.PerformQuery

SR_7040	GHTTPMessage.*, CIOProcessor.Input, CTranslator.sendGetJobList(J), CCommunicator.Send, DClientCommunicator.*, DClientTranslator.*, DClientManager.GetJobList(J), DDatabaseManager.PerformQuery
SR_7070	GJSDLDescription.*
SR_7110	See text above.
SR_7120	See text above.
SR_7130	See text above.
SR_7140	GJSDLDescription.*
SR_7150	See text above.
SR_8000	GHTTPMessage.*, AIOProcessor.Input, CIOProcessor.Input, CTranslator.sendLogIn, CCommunicator.Send, DIOProcessor.Input, DClientCommunicator.*, DClientTranslator.*, DClientManager.LogIn, DDatabaseManager.PerformQuery
SR_9000	See text above.
SR_9010	See text above.
SR_9020	See text above.
SR_9030	AIOProcessor.Input, CIOProcessor.Input, DIOProcessor.Input
SR_9031	See text above.
SR_9040	See text above.
SR_9050	See text above.
SR_9060	See text above.
SR_9070	See text above.
SR_9080	See text above.
SR_9090	See text above.
SR_9100	See text above.
SR_9110	See text above.
SR_9120	See text above.
SR_9130	GHTTPMessage.*
SR_9140	See text above.
SR_9150	See text above.
SR_9160	GLogger.Log, GLogger.Error

SR_9170	See text above.
SR_9180	See text above.

7.2 Architectural Design to Software Requirements

Architectural Design	**Software Requirements**
ACommunicator.send	SR_3061, SR_3010, SR_3040, SR_3070, SR_3071, SR_3020, SR_3021, SR_3050, SR_3051, SR_3060
ADistributor.sendDisapproveJobProvider	SR_3071
ADistributor.AddInterval	SR_3030
ADistributor.sendDisapproveApplicationProvider	SR_3061
ADistributor.sendPoll	SR_3010
ADistributor.sendApproveJobProvider	SR_3070
ADistributor.sendDisapproveApplication	SR_3051
ADistributor.sendGetUsing	SR_3040
ADistributor.sendRemoveProject	SR_3021
ADistributor.RemoveInterval	SR_3031
ADistributor.sendApproveApplication	SR_3050
ADistributor.sendAddProject	SR_3020
ADistributor.sendApproveApplicationProvider	SR_3060
AIOProcessor.Input	SR_3061, SR_3010, SR_3040, SR_3070, SR_3071, SR_3020, SR_3021, SR_3050, SR_3051, SR_9030, SR_3030, SR_3031, SR_3060, SR_8000
AJobScheduler.AddInterval	SR_3030
AJobScheduler.RemoveInterval	SR_3031
AJobScheduler.receivePoll	SR_3010
ATranslator.sendApproveJobProvider	SR_3070
ATranslator.sendAddProject	SR_3020
ATranslator.sendDisapproveApplication	SR_3051
ATranslator.sendRemoveProject	SR_3021
ATranslator.sendDisapproveJobProvider	SR_3071
ATranslator.sendGetUsing	SR_3040
ATranslator.sendDisapproveApplicationProvider	SR_3061
ATranslator.sendApproveApplication	SR_3050
ATranslator.sendPoll	SR_3010
ATranslator.sendApproveApplicationProvider	SR_3060

CCommunicator.Send	SR_6061, SR_4031, SR_5030, SR_4032, SR_4060, SR_4033, SR_2030, SR_2031, SR_2060, SR_2090, SR_6010, SR_6011, SR_6040, SR_4010, SR_7010, SR_4011, SR_7011, SR_7012, SR_4040, SR_7040, SR_5010, SR_7013, SR_2010, SR_4041, SR_5011, SR_4070, SR_5040, SR_4042, SR_2040, SR_2011, SR_4043, SR_2041, SR_5070, SR_5071, SR_2070, SR_6020, SR_2100, SR_6021, SR_6050, SR_4020, SR_7020, SR_4021, SR_4050, SR_5020, SR_2020, SR_4051, SR_5021, SR_2021, SR_5050, SR_2080, SR_6030, SR_6060, SR_4030, SR_7030, SR_8000
CIOProcessor.Input	SR_6061, SR_4031, SR_5030, SR_4032, SR_4060, SR_4033, SR_2030, SR_2031, SR_2060, SR_2090, SR_6010, SR_6011, SR_6040, SR_4010, SR_7010, SR_4011, SR_7011, SR_7012, SR_4040, SR_7040, SR_5010, SR_7013, SR_2010, SR_4041, SR_5011, SR_4070, SR_5040, SR_4042, SR_2040, SR_2011, SR_4043, SR_2041, SR_5070, SR_5071, SR_2070, SR_6020, SR_2100, SR_6021, SR_6050, SR_4020, SR_7020, SR_4021, SR_4050, SR_5020, SR_2020, SR_4051, SR_5021, SR_2021, SR_5050, SR_2080, SR_9030, SR_6030, SR_6060, SR_4030, SR_7030, SR_8000
CTranslator.sendGetResults	SR_7030
CTranslator.sendAllowOwnApplication	SR_4050
CTranslator.sendRemoveProjectAdmin	SR_2031
CTranslator.sendDisapproveResourceProvider	SR_4021
CTranslator.sendGetJobList(S)	SR_2060
CTranslator.sendApproveApplicationToProject	SR_5020
CTranslator.sendDisapproveDataProvider	SR_4033
CTranslator.sendGetApprovedDataProviders	SR_2090
CTranslator.sendGetApplicationList	SR_7020
CTranslator.sendAddDataProvider	SR_2020
CTranslator.sendRemoveJob(P)	SR_4060
CTranslator.sendAddProjectAdmin	SR_2030
CTranslator.sendApproveResourceProvider	SR_4020
CTranslator.sendAddData	SR_6010
CTranslator.sendGetApplicationsList(D)	SR_6050
CTranslator.sendGetTotalUsed	SR_5040
CTranslator.sendDisallowOwnApplication	SR_4051

CTranslator.sendDisapproveApplicationFromProject	SR_5021
CTranslator.sendRemoveDataProvider	SR_2021
CTranslator.sendApproveDataProvider	SR_4032
CTranslator.sendGetDataList	SR_6040
CTranslator.sendDisapproveDataFromProject	SR_6021
CTranslator.sendApproveJobProvider(P)	SR_4010
CTranslator.GetProjectList	SR_5050
CTranslator.sendOfferJob	SR_7010, SR_7012, SR_7013
CTranslator.sendGetApplicationsList(A)	SR_5030
CTranslator.sendGetResourceCalculations	SR_2100
CTranslator.sendDisapproveJobProvider(P)	SR_4011
CTranslator.sendApproveDataToProject	SR_6020
CTranslator.sendRemoveJob(J)	SR_7011
CTranslator.sendGetApprovedApplicationProviders	SR_2080
CTranslator.sendDisapproveDataset	SR_4031
CTranslator.sendGetProjectList	SR_6030
CTranslator.sendDisapproveJobProvider(D)	SR_6061
CTranslator.sendApproveApplication	SR_4040
CTranslator.sendLogIn	SR_8000
CTranslator.sendGetJobList(J)	SR_7040
CTranslator.sendGetProjectAdmins	SR_2070
CTranslator.sendAddApplication	SR_5010
CTranslator.sendDisapproveJobProvider(A)	SR_5071
CTranslator.sendAddApplicationProvider	SR_2010
CTranslator.sendApproveJobProvider(A)	SR_5070
CTranslator.sendApproveJobProvider(D)	SR_6060
CTranslator.sendApproveDataset	SR_4030
CTranslator.sendRemoveProject	SR_2041
CTranslator.sendDisapproveApplication	SR_4041
CTranslator.sendDisapproveApplicationProvider	SR_4043
CTranslator.sendRemoveApplication	SR_5011
CTranslator.sendRemoveApplicationProvider	SR_2011
CTranslator.sendGetJobList(P)	SR_4070
CTranslator.sendAddProject	SR_2040
CTranslator.sendApproveApplicationProvider	SR_4042
CTranslator.sendRemoveData	SR_6011
DAgentCommunicator.*	SR_3061, SR_3010, SR_3040, SR_3070, SR_3071, SR_3020, SR_3021, SR_3050, SR_3051, SR_3060
DAgentManager.DisapproveApplication	SR_3051
DAgentManager.Poll	SR_3010
DAgentManager.ApproveApplication	SR_3050
DAgentManager.GetUsing	SR_3040
DAgentManager.RemoveProject	SR_3021

DAgentManager.ApproveApplicationProvider	SR_3060
DAgentManager.AddProject	SR_3020
DAgentManager.ApproveJobProvider	SR_3070
DAgentManager.DisapproveApplicationProvider	SR_3061
DAgentManager.DisapproveJobProvider	SR_3071
DAgentTranslator.*	SR_3061, SR_3010, SR_3040, SR_3070, SR_3071, SR_3020, SR_3021, SR_3050, SR_3051, SR_3060
DClientCommunicator.*	SR_6061, SR_4031, SR_5030, SR_4032, SR_4060, SR_4033, SR_2030, SR_2031, SR_2060, SR_2090, SR_6010, SR_6011, SR_6040, SR_4010, SR_7010, SR_4011, SR_7011, SR_7012, SR_4040, SR_7040, SR_5010, SR_7013, SR_2010, SR_4041, SR_5011, SR_4070, SR_5040, SR_4042, SR_2040, SR_2011, SR_4043, SR_2041, SR_5070, SR_5071, SR_2070, SR_6020, SR_2100, SR_6021, SR_6050, SR_4020, SR_7020, SR_4021, SR_4050, SR_5020, SR_2020, SR_4051, SR_5021, SR_2021, SR_5050, SR_2080, SR_6030, SR_6060, SR_4030, SR_7030, SR_8000
DClientManager.GetApprovedApplicationProviders	SR_2080
DClientManager.ApproveJobProvider(A)	SR_5070
DClientManager.RemoveProject	SR_2041
DClientManager.AddApplication	SR_5010
DClientManager.DisapproveJobProvider(A)	SR_5071
DClientManager.GetJobList(J)	SR_7040
DClientManager.GetProjectAdmins	SR_2070
DClientManager.LogIn	SR_8000
DClientManager.ApproveApplication	SR_4040
DClientManager.GetTotalUsed	SR_5040
DClientManager.ApproveJobProvider(D)	SR_6060
DClientManager.GetResults	SR_7030
DClientManager.DisapproveApplication	SR_4041
DClientManager.RemoveJob(J)	SR_7011
DClientManager.RemoveApplicationProvider	SR_2011
DClientManager.DisapproveJobProvider(P)	SR_4011
DClientManager.ApproveDataToProject	SR_6020
DClientManager.RemoveApplication	SR_5011
DClientManager.GetProjectList(D)	SR_6030
DClientManager.GetResourceCalculations	SR_2100
DClientManager.OfferJob	SR_7010, SR_7012, SR_7013
DClientManager.ApproveJobProvider(P)	SR_4010
DClientManager.AddProject	SR_2040

DClientManager.GetDataList	SR_6040
DClientManager.DisapproveDataToProject	SR_6021
DClientManager.DisapproveJobProvider(D)	SR_6061
DClientManager.DisapproveDataSet	SR_4031
DClientManager.GetJobList(P)	SR_4070
DClientManager.RemoveData	SR_6011
DClientManager.ApproveApplicationProvider	SR_4042
DClientManager.AddData	SR_6010
DClientManager.GetApplicationList(D)	SR_6050
DClientManager.DClientTranslator.*	SR_4042
DClientManager.ApproveResourceProvider	SR_4020
DClientManager.GetApprovedDataProviders	SR_2090
DClientManager.GetApplicationList(J)	SR_7020
DClientManager.AddApplicationProvider	SR_2010
DClientManager.DisapproveApplicationToProject	SR_5021
DClientManager.GetJobList(S)	SR_2060
DClientManager.ApproveDataSet	SR_4030
DClientManager.GetProjectList(A)	SR_5050
DClientManager.RemoveProjectAdmin	SR_2031
DClientManager.DisapproveResourceProvider	SR_4021
DClientManager.GetApplicationList(A)	SR_5030
DClientManager.AddProjectAdmin	SR_2030
DClientManager.AllowOwnApplication	SR_4050
DClientManager.DisapproveDataProvider	SR_4033
DClientManager.ApproveApplicationToProject	SR_5020
DClientManager.RemoveJob(P)	SR_4060
DClientManager.AddDataProvider	SR_2020
DClientManager.RemoveDataProvider	SR_2021
DClientManager.ApproveDataProvider	SR_4032
DClientManager.DisallowOwnApplication	SR_4051
DClientManager.DisapproveApplicationProvider	SR_4043
DClientTranslator.*	SR_6061, SR_4031, SR_5030, SR_4032, SR_4060, SR_4033, SR_2030, SR_2031, SR_2060, SR_2090, SR_6010, SR_6011, SR_6040, SR_4010, SR_7010, SR_4011, SR_7011, SR_7012, SR_4040, SR_7040, SR_5010, SR_7013, SR_2010, SR_4041, SR_5011, SR_4070, SR_5040, SR_2040, SR_2011, SR_4043, SR_2041, SR_5070, SR_5071, SR_2070, SR_6020, SR_2100, SR_6021, SR_6050, SR_4020, SR_7020, SR_4021, SR_4050, SR_5020, SR_2020, SR_4051, SR_5021, SR_2021, SR_5050, SR_2080, SR_6030, SR_6060, SR_4030, SR_7030, SR_8000

DDatabaseManager.PerformQuery	SR_3061, SR_6061, SR_4031, SR_5030, SR_4032, SR_4060, SR_4033, SR_2030, SR_2031, SR_2060, SR_2090, SR_3010, SR_6010, SR_6011, SR_3040, SR_6040, SR_4010, SR_7010, SR_4011, SR_7011, SR_7012, SR_3070, SR_4040, SR_7040, SR_5010, SR_7013, SR_2010, SR_3071, SR_4041, SR_5011, SR_4070, SR_5040, SR_4042, SR_2040, SR_2011, SR_4043, SR_2041, SR_5070, SR_5071, SR_2070, SR_3020, SR_6020, SR_2100, SR_3021, SR_6021, SR_3050, SR_6050, SR_4020, SR_7020, SR_3051, SR_4021, SR_4050, SR_5020, SR_2020, SR_4051, SR_5021, SR_2021, SR_5050, SR_2080, SR_6030, SR_3060, SR_6060, SR_4030, SR_7030, SR_8000
DIOProcessor.Input	SR_9030, SR_8000
GHTTPMessage.*	SR_3061, SR_6061, SR_4031, SR_5030, SR_4032, SR_4060, SR_4033, SR_2030, SR_2031, SR_2060, SR_2090, SR_3010, SR_6010, SR_3011, SR_6011, SR_3040, SR_6040, SR_4010, SR_7010, SR_4011, SR_7011, SR_7012, SR_3070, SR_4040, SR_7040, SR_5010, SR_7013, SR_2010, SR_3071, SR_4041, SR_5011, SR_4070, SR_5040, SR_4042, SR_2040, SR_2011, SR_4043, SR_9130, SR_2041, SR_5070, SR_5071, SR_2070, SR_3020, SR_6020, SR_2100, SR_3021, SR_6021, SR_3050, SR_6050, SR_4020, SR_7020, SR_3051, SR_4021, SR_4050, SR_5020, SR_2020, SR_4051, SR_5021, SR_2021, SR_5050, SR_2080, SR_3030, SR_6030, SR_3031, SR_3060, SR_6060, SR_4030, SR_7030, SR_8000
GJSDLDescription.GetCPUArchitecture	SR_1414
GJSDLDescription.GetGetJobAnnotations	SR_1230
GJSDLDescription.GetJobDescription	SR_1220
GJSDLDescription.GetTDiskSpace	SR_1426
GJSDLDescription.GetICPUTime	SR_1416
GJSDLDescription.GetIVirtualMemory	SR_1420
GJSDLDescription.GetIDiskSpace	SR_1421
GJSDLDescription.GetDataStaging	SR_1510, SR_1540, SR_1520, SR_1550, SR_1530, SR_1560

GJSDLDescription.GetTCPUTime	SR_1422
GJSDLDescription.GetTCPUCount	SR_1423
GJSDLDescription.GetICPUSpeed	SR_1415
GJSDLDescription.GetApplicationDescription	SR_1330
GJSDLDescription.GetICPUCount	SR_1417
GJSDLDescription.GetApplicationVersion	SR_1320
GJSDLDescription.GetOperatingSystem	SR_1413
GJSDLDescription.GetINetworkBandwidth	SR_1418
GJSDLDescription.GetApplicationName	SR_1310
GJSDLDescription.GetTPhysicalMemory	SR_1424
GJSDLDescription.GetExclusiveExecution	SR_1412
GJSDLDescription.GetFileSystems	SR_1411
GJSDLDescription.GetCandidateHosts	SR_1410
GJSDLDescription.GetGetJobProjects	SR_1240
GJSDLDescription.GetJobName	SR_1210
GJSDLDescription.GetTResourceCount	SR_1427
GJSDLDescription.*	SR_7070, SR_7140
GJSDLDescription.GetIPhysicalMemory	SR_1419
GJSDLDescription.GetTVirtualMemory	SR_1425
GLogger.Error	SR_9160
GLogger.Log	SR_9160

Appendix A

Communication protocol

The protocol describes how communication will take place between the different software programs. The communication is done using HTTP POST requests sent by the agent en client to the dispatcher. The dispatcher then responds with a HTTP response. The protocol should be implemented by the HTTP Message component which is present in the client, agent and dispatcher. We use this kind of communication, because this would allow the client and agent to operate behind a firewall or NAT configuration. An HTTP request is actually a string containing pairs of variables and values and a HTTP response contains a HTTP status code and a string.

A.1 Client ↔ Dispatcher

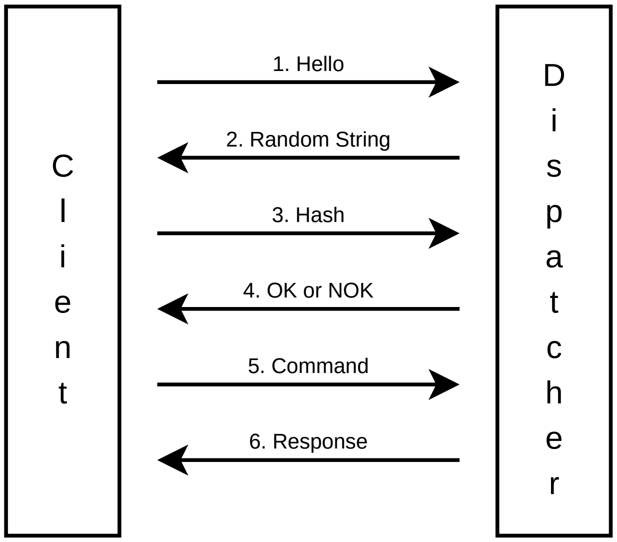

Figure A.1: Protocol between Client and Dispatcher

The communication is graphically shown in figure A.1. The client can send three types of packets to the dispatcher:

- Register-packet (see section **??**)
- Hello-packet (see section **??**)
- Hash-packet (see section **??**)
- Command-packet (see section A.1.3)

A.1.1 Register

Before a user can use the services provided by the SPINGRID system, it needs to register itself with the dispatcher. It does by sending a register-packet to the dispatcher:

Variable	**Value**
command	Register
username	(username)
password	(password)

The fields *username* and *password* contain the desired username and password for the user.

A.1.2 Log In

Before the client can perform his actions, it needs to log in. It does this by sending an hello-packet to the dispatcher:

Variable	**Value**
command	Login_Ask
username	(username)

The dispatcher generates a random string and sends this back to the client as an HTTP response (1). Both the dispatcher and the client then use the random string and the username of the user operating the client to calculate a hash. This means that the dispatcher should know all passwords of all users. The client then sends his hash back to the dispatcher in a hash-packet:

Variable	**Value**
command	Login_Hash
username	(username)
hash	(hash)

The dispatcher compares its own hash with the received hash. If the hashes are equal the client is considered authenticated and a ”LoginOK” string is returned. If the hashes are not equal a ”LoginNOK” is returned.

A.1.3 Send commands

After the client has logged in, it can send commands to the dispatcher. A command-packet has the form of

Variable	**Value**
command	(command)
username	(username)
hash	(hash)
arg1	(arg1)
arg2	(arg2)
...	...
argn	(argn)

The dispatcher then responds containing a HTTP status code and a string. The client can take action depending on the HTTP status code (see section A.3).

A.2 Agent ↔ Dispatcher

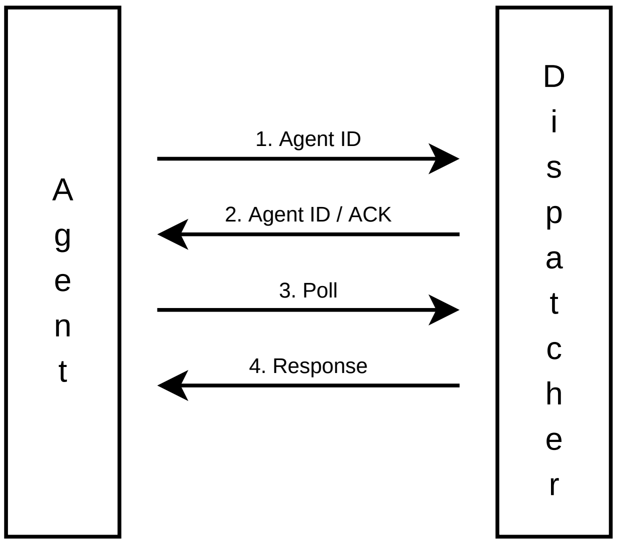

Figure A.2: Protocol between Agent and Dispatcher

In every instance of communication the client will send a string using HTTP POST and the dispatcher will then send a response. There is no situation in which the dispatcher initiates the communication. The packets have the same form as in the client protocol (see section A.1).

A.2.1 Identification

Before the agent can get a job, it needs to identify itself in the SPINGRID system. It does this by sending its *AgentID* and specifications to the dispatcher. If the agent is new, it sends an empty *AgentID*:

Variable	Value
command	Identify
agentid	*null*
specs	(specifications)

The dispatcher responds with a new unique AgentID. After the agent has received the new id, it can use it to identify itself. After identification the agent can send commands in the same way as the client does.

A.2.2 Polling

On regular intervals, the agent needs to poll the dispatcher, notifying it of its presence. A polling packet has the form of:

Variable	Value
command	Poll
agentid	(agentid)
status	(status)

The status can be either asking for a job (asking), just completed a job (completed) or currently calculating on a job (calculating). The dispatcher will respond with an HTTP response.

A.3 HTTP Status codes

In the table below an enumeration of all HTTP status codes that can be sent by the dispatcher is found.

HTTP Code	Meaning
200	The command has a valid syntax
202	The command was accepted
207	The command was not accepted
400	The command was not recognized or wrong parameters given
401	The command was not executed, because the client was not properly authenticated
404	Not found
500	While the dispatcher was executing the command, an unknown error occurred.

A.4 MySQL Results

Sometimes the dispatcher needs to return the result of a query. This query is in the form of a table, but it is only possible to send strings. Therefor, the tables have to be converted to a string and back. This is done in the following way:

Column_1	**Column_2**	**...**	**Column_n**
Field_1,1	Field_1,2	...	Field_1,n
Field_2,1	Field_2,2	...	Field_2,n
...	...	...	...
Field_m,1	Field_m,2	...	Field_m,n

is equivalent to the string:

```
Column_1%Column_2%...%Column_n%%Field_1,1%Field_1,2%...%Field_1,n%Field_2,1
%Field_2,2%...%Field_2,n%...%Field_m,1%Field_m,2%...%Field_m,n
```

Appendix B

Command line syntax

B.1 Dispatcher syntax

In this section the command-line syntax of the dispatcher application is described. To start the dispatcher use:

```
java -jar sgdispatcher.jar
```

The program will wait for one of the following commands:

- `exit`
  Shuts down the dispatcher application.
- `kill`
  Terminates the dispatcher application instantly, without shutting down individual components.

B.2 Client syntax

In this section the command-line syntax of the client application is described. The general form of a command is:

```
java -jar sgclient.jar <command>
```

where `<command>` contains an action followed by a set of arguments described below. In the description, variables are placed between `<` and `>`. For example, `<user>` means that a username is expected. All identifiers should be prefixed with an abbreviation of their type. For example, if an application needs to be passed, the argument could be `app:appname.exe`.

Variable explanation:

Variable	**Explanation**	**Type Abbr.**	**Example**
`<application>`	an application name	`app`	`app:someapp`
`<data>`	a dataset	`dat`	`dat:gaiadata001`
`<job>`	a number of a job	`job`	`job:23661`
`<project>`	a project	`proj`	`proj:GaiaProject`
`<role>`	a role	`rol`	`rol:dataprov`
`<url>`	a URL to a dataset, application or JSDL-file	`url`	`url:http://www.dom.com/j.jsdl`
`<user>`	a username	`usr`	`usr:henk`

Different roles:

`appprov`	application provider
`datprov`	data provider
`jobprov`	job provider
`projadmin`	project admin
`sysadmin`	system admin

Possible Commands:

- `add {app|data|job|project} {<url>|<project>} [<user>|<name>|<name> <url>]`
  Adds an application, a dataset, a project or a job to the system. If a project is added a username should be given as well to add an initial project manager to the system. If an application is added to the system, a name for that application should be given as well as the requirements file for it. If a dataset is added a name should be given for it.

- `approve [<role>] {<user>|<application>|<data>} for {<project>|<user>}`
  Approves a job provider to provide jobs in a project, a resource provider to provide his resource to a project, a dataset to be used in a project, a data provider to provide data in a project, an application to be used in a project, an application provider to provide applications for the project or a project admin in a given project. A user of the client program can have multiple roles which can lead to an ambiguous interpretation of the approve keyword. In this case the user must provide his role after the approve keyword.

- `approve private {dat|app} for <username> in <project>`
  Approve a user to provide his own applications or datasets in his jobs in a certain project.

- `assign <role> to <user>`
  Assigns a role to a user.

- `disapprove [<role>] {<user>|<application>|<data>} for {<project>|<user>}`
  Disapproves a job provider to provide jobs in a project, a resource provider to provide his resource to a project, a dataset to be used in a project, a data provider to provide data in a project, an application to be used in a project, an application provider to provide applications for the project. A user of the client program can have multiple roles which can lead to an ambiguous interpretation of the disapprove keyword. This user must provide his role after the disapprove keyword.

- `disapprove private {application|data} for <username> in <project>`
  Disapprove a user to provide his own applications or datasets in his jobs in a certain project.
- `list approved <role> in <project>`
  List approved application providers in a project or approved data providers in a project
- `list [<role>] apps [using <data>]`
  List all applications in the system (and how often they were used), all applications of the user, all applications available to the user or all applications that use a specific dataset.
- `list data`
  List all datasets the user has added.
- `list [<role>] jobs [in <project>]`
  List all jobs the user has added, all jobs on the system or all jobs in the system.
- `list projects using {<resource>|{apps|data}`
  List all projects using a specific resource or list all projects that use the apps/data of the user.
- `list users having <role>`
  Returns a list of users that have the role of `<role>`
- `list result <job>`
  Returns the result of a job.
- `login <username> <password>`
  Logs the user into the system
- `register <username> <password>`
  Registers the user into the system with the given password.
- `remove {<user>|<application>|<data>|<project>|<job>}`
  Removes a user, an application, a dataset, a project or a job from the system.
- `unassign <role> [from] <username>`
  Unassign a role from a user

B.3 Agent syntax

In this section the command-line syntax of the client application is described. To start the agent use:

```
java -jar sgagent.jar
```

The program will wait for one of the following commands:

- `add interval <time> <time>`
  Adds an interval in which the resource provides its services to the dispatcher.
- `approve {<application>|<user>|<project>}`
  Trust an application on the resource, trust all applications from an application provider or trust all jobs from a user on the resource or trust a project on your resource.
- `disapprove {<application>|<user>}`
  Distrust an application on the resource, distrust all applications from an application provider or distrust all jobs from a user on your resource.
- `exit`
  Shuts down the agent application.
- `kill`
  Terminates the agent application instantly, without shutting down individual components.
- `list using`
  List all projects that are using or have been using the resource.
- `remove interval <time> <time>`
  Removes an interval in which the resource provides its services to the dispatcher.