In Tryllian ADK habitat is analogue of container in Jade. The building blocks of the habitat are agents. The XML habitat description files allow you to specify which of these components you want to create. There is only one habitat per VM. Habitat is merely a word for the agent container: a collection of agents that provides services.

The XML file for the Hello World example is as follows:

<!DOCTYPE habitat SYSTEM "http://www.tryllian.com/xml/habitat\_1\_1.dtd">

<habitat>

<agent dna="scenarios/agents/helloworld.dna"/>

</habitat>

The starting point of every agent is a Java class that extends the Agent interface from the tryllian.afc.agent package. It’s similar to jade.core.Agent class in Jade.

The code of the HelloWorldAgent:

package tryllian.scenarios.helloworld;

import tryllian.afc.agent.Agent;

import tryllian.afc.task.standard.LogTask;

import tryllian.afc.task.standard.PeriodicalTask;

public class HelloWorldAgent extends Agent

implements tryllian.are.TransientAgent

{

public HelloWorldAgent(String[] args) {

// Set name if specified in habitat xml file

if (args != null && args.length > 0) {

setName(args[0]);

}

else {

setName("hello-agent");

}

addTask(new PeriodicalTask(new LogTask("Hello World!")), 30);

}

}

The structure of the agent is similar to the agent in Jade. We also extend Agent class. But in Trillian ADK initialization is performed inside constructor and we add tasks, not behaviours like in Jade. Periodical task is analog of TickerBehaviour in Jade.

In Trillian ADK se need to pack agents into DNA files, in Jade we simly use class files. The DNA file is a file that contains:

* The jar file of your agent
* The optional libraries that your agent uses
* A file descriptor that contains the location of the class file of your agent in the jar file and the location of the libraries needed
* The signature of the agent with your private key for authentification and versioning

The call to ANT is wrapped in a shell script or a batch file, depending upon the host OS. This makes recreating agents very simple: to compile the Hello World example and create the agent, type:

**build helloworld**

The build process consists of the following steps:

1. Compiling the Java classes.
2. Packaging the class files in one or more jar file.
3. Creating the agent dna file (combining a descriptor file and jar files together).

After building, a file called helloworld.dna will be built from your new code in the agents directory. To run your agent, type

**habitat helloworld.xml**

In Jade we just print two lines:

javac HelloAgent.java

java jade.Boot fred:HelloAgent

To register registers the agent with the specified service name in Tryllian we can use RegisterTask class. This isanalogue of DFService.register in Jade.

To find agents in Tryllian ADK we can use tryllian.afc.task.find.Directory interface:

* FindAgentNamesTask retrieves the names of all agents in a habitat.
* FindAgentsTask retrieves the addresses of all agents in a habitat.
* FindHabitatServiceTask retrieves the address of the Habitat Service Agent.
* JNDIResolveTask resolves a jndi context-name.
* QueryAgentNameTask asks an agent for its name.
* TrackAgentsTask is notified when an agent arrives or departs.
* **getServiceNames** returns a set of all currently registered services.

The Directory interface is analogue to DFService.search in Jade

Jade Projects

**LEAP project**

During the 2000 – 2002 time span, thanks to the valuable contribution of the Consortium partners - Motorola (coordinator), ADAC, Broadcom, BT, Telecom Italia, University of Parma, Siemens – the project accomplished the relevant goal of developing a new lightweight runtime environment for JADE that can be both exploited in the Internet and in the wireless context. The LEAP extension allows the deployment of JADE on J2ME devices, and open the doors for mobile applications based on the P2P intelligent agent approach.

[TeSCHeT](http://www.teschet.net/)  
**Timeframe:**   
2002-2005  
**Participants:**  
Telecom Italia, Engineering Ingegneria Informatica S.p.A, Isufi  
**Goal:**  
The project intends to study and develop a platform-demonstrator, characterized by "open source" standards and technologies, and based on the JADE platform, enabling the treatment and the organization of the tourist-cultural information and the creation of spontaneous services networks

[PRIMO](http://primo.ismb.it/)  
**Timeframe:**   
2002-2005  
**Participants:**  
CERCOM - POLITECNICO DI TORINO, DEPARTMENT OF INFORMATION ENGINEERING OF THE UNIVERSITY OF PISA, DEPARTMENT OF INFORMATION ENGINEERING OF THE UNIVERSITY OF PADOVA, ISTITUTO SUPERIORE MARIO BOELLA, DEPARTMENT OF SCIENCE AND TECHNIQUE OF COMMUNICATIONS AND INFORMATION (INFOCOM) OF THE UNIVERSITY OF ROMA "LA SAPIENZA", DEPARTMENT OF ELECTRONIC ENGINEERING AND INFORMATION (DIEI) OF THE UNIVERSITÀ OF PERUGIA, DEPARTMENT OF ELECTRIC ENGINNERING (DIE) OF THE UNIVERSITY OF PALERMO¸ DEPARTMENT OF ELECTRIC ENGINNERING (DIE) OF THE UNIVERSITY OF ROMA “TOR VERGATA”, STMICROELECTRONICS, TELECOM ITALIA LAB, MARCONI SELENIA COMMUNICATIONS  
**Goal:**  
The PRIMO project deals with the design, the development, and the experimental validation of base stations and user terminals for wideband wireless communications systems able to cope with those reconfigurability and interoperability characteristics required by the next generation mobile communication systems.

[\*E-Commerce Agent Platform (E-CAP)\*](http://e-cap.sourceforge.net/)  
 **Timeframe:**   
2006-2008  
**Participants:**Systems Research Institute of Polish Academy of Sciences  
**Goal:**Comprehensive agent-based e-commerce system realizing price negotations and combining: (a) adaptability and intelligence (change strategy according to history of purchases), (b) mobility (based on transfer of negotation modules and agents). Project is supported by a Marie Curie IRG Grant

Tryllian ADK Projects

1. WORKSTATION MANAGEMENT

At a big five consulting firm, all of the employees have administrator rights at their workstations. As a result of this each person can install software on his or her workstation. When this software is not licensed with the company it can result in license infractions.

Through the use of Tryllian’s workstation agent application this type of scenario can be avoided. Not only can Tryllian’s workstation application detect certain software but it can also perform diagnostic du­ties such as updates on software that is allowed on each employee’s workstation. All of the components that make up the application are put into use on a closed network, with the possibility of being extended beyond the firewall for home workers and consultants.

2. AGENT-BASED SCHEDULING

Through the use of ADK building blocks an agent application has been created to integrate and simplify the communication and work process in the Rotterdam harbor.

In this process four departments are involved: forwarding, the customs department, the transport services and the International Transport Service (ETT). The designed and implemented workflow architecture demonstrates how agent applications can be used to integrate and create efficiency in com­plicated situations.

The situation is as follows; containers are continually arriving in the Rotterdam harbor and need to be checked by the customs department and then scheduled for transport to their final destinations. This is done through a series of communications by different departments throughout the harbor by fax and telephone. This is an outdated and slow procedure. By using agent applications all of the involved parties were able to integrate their actions and have the results returned to them by agents. By remov­ing overlapping and confusing communication this not only speeded up the process, but created time frames where other duties could be performed.