Master Thesis: Single Sign-On (JOSSO)

Tzani Adamantia (MTE:1068)

Supervisor Professor: Dr. Xenakis Christos

December 2012
Contents

1. Introduction ..................................................................................................................... 11
2. What is JOSSO ................................................................................................................ 11
3. JOSSO Editions ............................................................................................................... 13
4. JOSSO Versions .............................................................................................................. 14
5. Provision and analysis of High-availability and Scalability ............................................ 15
   5.1 System Failover ....................................................................................................... 15
   5.2 Session Failover ...................................................................................................... 15
   5.3 Failover Mechanism Architecture ........................................................................... 15
   5.4 Enabling Clustering ................................................................................................. 17
   5.5 Configuration Replication ....................................................................................... 17
   6.1 Creating a server certificate with keytool ................................................................ 18
   6.2 Enabling SSL in the Web Server used by JOSSO ................................................... 19
   6.3 Exporting Web Server SSL Certificate ................................................................... 19
   6.4 Agents connection to Web Services over SSL in JOSSO ....................................... 20
7. High Level Architecture of JOSSO ................................................................................. 21
8. Activity Diagram (State Machine) .................................................................................. 24
9. How to install JOSSO2 .................................................................................................... 25
10. Installation steps for JOSSO2 ...................................................................................... 26
11. Identity Appliance ....................................................................................................... 38
12. Entities ......................................................................................................................... 41
13. Configuration and installation of the Identity Provider’s Identity Store ....................... 42
14. Execution Environments ............................................................................................. 43
15. Configuration and installation of the Service Provider’s Execution Environment ..... 45
    15.1 Alfresco Execution Environment ......................................................................... 45
    15.2 Apache Web Server Execution Environment ....................................................... 46
    15.3 Java EE Execution Environment .......................................................................... 48
    15.4 JBoss Portal Execution Environment .................................................................. 49
    15.5 Liferay Portal Execution Environment ............................................................... 50
    15.6 PhpBB Execution Environment ......................................................................... 51
    15.7 Webserver Execution Environment .................................................................... 52
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.8</td>
<td>Oracle Weblogic Execution Environment</td>
</tr>
<tr>
<td>15.9</td>
<td>Websphere Community Edition (WASCE) Execution Environment</td>
</tr>
<tr>
<td>15.10</td>
<td>Windows IIS Execution Environment</td>
</tr>
<tr>
<td>15.11</td>
<td>Apache Tomcat Execution Environment</td>
</tr>
<tr>
<td>15.12</td>
<td>JBoss Execution Environment</td>
</tr>
<tr>
<td>16.</td>
<td>Authentication</td>
</tr>
<tr>
<td>17.</td>
<td>Authentication Setup</td>
</tr>
<tr>
<td>17.1</td>
<td>Configuration an installation of Directory-based Authentication</td>
</tr>
<tr>
<td>17.2</td>
<td>Configuration and installation of Integrated Windows Authentication</td>
</tr>
<tr>
<td>17.3</td>
<td>Configuration and installation Two-Factor Authentication</td>
</tr>
<tr>
<td>18.</td>
<td>Identity Sources</td>
</tr>
<tr>
<td>18.1</td>
<td>Configuration and installation of an Identity Source</td>
</tr>
<tr>
<td>18.2</td>
<td>Configuration and installation of an Identity Vault</td>
</tr>
<tr>
<td>18.3</td>
<td>Configuration and installation of an LDAP Directory Identity Source</td>
</tr>
<tr>
<td>18.4</td>
<td>Configuration and installation of a RDBMS Identity Source</td>
</tr>
<tr>
<td>18.5</td>
<td>Configuration and installation of a XML Identity Source</td>
</tr>
<tr>
<td>19.</td>
<td>Connections</td>
</tr>
<tr>
<td>19.1</td>
<td>Federated connection</td>
</tr>
<tr>
<td>19.2</td>
<td>Activation</td>
</tr>
<tr>
<td>19.3</td>
<td>Identity Lookup</td>
</tr>
<tr>
<td>19.4</td>
<td>Identity Verification</td>
</tr>
<tr>
<td>20.</td>
<td>Privilege and Account Management</td>
</tr>
<tr>
<td>20.1</td>
<td>User Creation</td>
</tr>
<tr>
<td>20.2</td>
<td>How to create a Group</td>
</tr>
<tr>
<td>20.3</td>
<td>How to add an extra attribute</td>
</tr>
<tr>
<td>21.</td>
<td>Identity Appliance Lifecycle Management Screen</td>
</tr>
<tr>
<td>22.</td>
<td>System Setting</td>
</tr>
<tr>
<td>23.</td>
<td>Creating an Identity Appliance for SSO</td>
</tr>
<tr>
<td>24.</td>
<td>Cloud computing in JOSSO</td>
</tr>
<tr>
<td>25.</td>
<td>Two-Factor Authentication with WiKID</td>
</tr>
<tr>
<td>26.</td>
<td>Conclusion</td>
</tr>
<tr>
<td>27.</td>
<td>References</td>
</tr>
</tbody>
</table>
Pictures

Picture.1 Failover Mechanism Architecture ................................................................. 16
Picture.2 High Level Architecture of JOSSO ................................................................. 21
Picture.3 State Machine .................................................................................................. 24
Picture.4 Install JOSSO .................................................................................................... 27
Picture.5 Running JOSSO ................................................................................................. 28
Picture.6 osgi:list | grep Atricore ..................................................................................... 29
Picture.7 Loading JOSSO ................................................................................................. 30
Picture.8 Login Page ....................................................................................................... 31
Picture.9 Step1 of JOSSO EE Installation ...................................................................... 31
Picture.10 Step 2 of JOSSO EE Installation .................................................................... 32
Picture.11 Step 3 of JOSSO EE Installation .................................................................... 32
Picture.12 Step 4 of JOSSO EE Installation .................................................................... 33
Picture.13 Step 5 of JOSSO EE Installation .................................................................... 33
Picture.14 Step 6 of JOSSO EE Installation .................................................................... 34
Picture.15 Step 7 of JOSSO EE Installation .................................................................... 34
Picture.16 Activation License Window ............................................................................. 35
Picture.17 Identity Appliance Modeler Screen ............................................................... 36
Picture.18 Administrator Window .................................................................................... 36
Picture.19 Change Password .......................................................................................... 37
Picture.20 Five tabs of Atricore ...................................................................................... 37
Picture.21 Alfresco Execution Environment ................................................................. 45
Picture.22 Apache Web Server Execution Environment ................................................. 47
Picture.23 Java EE Execution Environment ..................................................................... 48
Picture.24 JBoss Portal Execution Environment ............................................................. 49
Picture.25 Liferay Portal Execution Environment .......................................................... 50
Picture.26 PhpBB Execution Environment ...................................................................... 51
Picture.27 Webserver Execution Environment ............................................................... 53
Picture.28 Oracle Weblogic Execution Environment ....................................................... 54
Picture.29 Websphere Community Edition (WASCE) Execution Environment ............. 55
Picture.30 Windows IIS Execution Environment ............................................................. 56
Picture.31 Apache Tomcat Execution Environment ........................................................ 57
Picture.32 JBoss Execution Environment ......................................................................... 59
Picture.33 Authentication Base tab ................................................................. 61
Picture.34 Authentication Lookup tab ............................................................. 62
Picture.35 Windows Authentication ................................................................... 64
Picture.36 WiKID Authentication ...................................................................... 66
Picture.37 Create Identity Vault ........................................................................ 68
Picture.38 Create LDAP Identity Source Base tab .............................................. 69
Picture.39 Create LDAP Identity Source Lookup tab .......................................... 70
Picture.40 Create DB Identity Source Connection tab ....................................... 72
Picture.41 Create DB Identity Source User Lookup tab ....................................... 73
Picture.42 Create DB Identity Source Password Update tab ................................ 74
Picture.43 Create XML Identity Source ............................................................... 78
Picture.44 Account & Entitlement Management Screen ..................................... 80
Picture.45 Manage Users .................................................................................... 80
Picture.46 Manage Groups .................................................................................. 81
Picture.47 Create User1 ...................................................................................... 81
Picture.48 Create User Button ............................................................................ 81
Picture.49 Create User General tab ................................................................. 82
Picture.50 Create User Preferences tab .............................................................. 83
Picture.51 Create User Group 1 tab ................................................................... 83
Picture.52 Create Group tab ............................................................................... 84
Picture.53 Create User Password tab ............................................................... 85
Picture.54 Save Button ...................................................................................... 85
Picture.55 Account & Entitlement Management 1 Screen ................................. 86
Picture.56 Delete User ...................................................................................... 86
Picture.57 Confirmation of Deletion Window ..................................................... 87
Picture.58 Edit User Button ............................................................................... 87
Picture.59 Edit User Window ............................................................................ 88
Picture.60 Search for a User ............................................................................. 88
Picture.61 Account & Entitlement Management 2 Screen ................................. 89
Picture.62 Remove Search Filters Button .......................................................... 89
Picture.63 Create Group .................................................................................... 90
Picture.64 Create Group1 .................................................................................. 91
Picture.65 Edit Group Window ........................................................................... 91
Picture.66 Delete Group ................................................................................... 92
Picture.67 Confirms Deletion of a Group Window ................................................................. 92
Picture.68 Search Group Window ........................................................................................... 93
Picture.69 Search Group Results ............................................................................................. 93
Picture.70 Add an Extra Attribute ........................................................................................... 94
Picture.71 Manage Schemas .................................................................................................... 94
Picture.72 Add Schemas Attribute .......................................................................................... 95
Picture.73 Extra Attribute ........................................................................................................ 95
Picture.74 Identity Appliance Lifecycle Management Screen ................................................ 96
Picture.75 Moving to the next Column ................................................................................... 97
Picture.76 Staged Column ....................................................................................................... 97
Picture.77 Starting appliance ................................................................................................... 98
Picture.78 Deployed Column .................................................................................................. 98
Picture.79 Return to the Staged Column ................................................................................. 99
Picture.80 Disposed Column ................................................................................................. 100
Picture.81 Delete the Identity Appliance ................................................................................ 100
Picture.82 Identity Appliance Lifecycle Management 1 ....................................................... 100
Picture.83 System Settings .................................................................................................... 101
Picture.84 E-mail notification tab .......................................................................................... 101
Picture.85 SMTP tab ............................................................................................................. 102
Picture.86 Action Bar ............................................................................................................ 103
Picture.87 Create the Identity Appliance ............................................................................... 103
Picture.88 Confirm Identity Appliance Window ................................................................... 104
Picture.89 Identity Appliance Modeler Window ................................................................... 104
Picture.90 Entities ................................................................................................................. 105
Picture.91 Identity Provider Care tab .................................................................................... 105
Picture.92 Identity Provider Contract tab .............................................................................. 107
Picture.93 Identity Provider Certificate tab ........................................................................... 108
Picture.94 Identity Appliance Modeler with Identity Provider ............................................. 109
Picture.95 Identity Vault ....................................................................................................... 110
Picture.96 Create Identity Vault ............................................................................................ 110
Picture.97 Identity Lookup Connection ................................................................................ 110
Picture.98 Service Provider ................................................................................................... 111
Picture.99 Create a Service Provider Core tab ...................................................................... 111
Picture.100 Create a Service Provider Contract tab .............................................................. 113
### Single Sign-On

<table>
<thead>
<tr>
<th>Picture Ref</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Create a Service Provider Certificate tab</td>
<td>114</td>
</tr>
<tr>
<td>102</td>
<td>Create Tomcat</td>
<td>116</td>
</tr>
<tr>
<td>103</td>
<td>Create JBoss</td>
<td>116</td>
</tr>
<tr>
<td>104</td>
<td>Activation</td>
<td>117</td>
</tr>
<tr>
<td>105</td>
<td>Activate Tomcat</td>
<td>117</td>
</tr>
<tr>
<td>106</td>
<td>Activate JBoss</td>
<td>118</td>
</tr>
<tr>
<td>107</td>
<td>Reactivate window</td>
<td>118</td>
</tr>
<tr>
<td>108</td>
<td>Users and Groups</td>
<td>119</td>
</tr>
<tr>
<td>109</td>
<td>Bring Identity Appliance into Life Saved Column</td>
<td>119</td>
</tr>
<tr>
<td>110</td>
<td>Bring Identity Appliance into Life Deployed Column</td>
<td>120</td>
</tr>
<tr>
<td>111</td>
<td>Atricore Login window</td>
<td>120</td>
</tr>
<tr>
<td>112</td>
<td>Fill in Username and Password</td>
<td>121</td>
</tr>
<tr>
<td>113</td>
<td>Loaded the Login Page</td>
<td>121</td>
</tr>
<tr>
<td>114</td>
<td>SSO</td>
<td>122</td>
</tr>
<tr>
<td>115</td>
<td>If we click on “protected” option the following picture appears.</td>
<td>122</td>
</tr>
<tr>
<td>116</td>
<td>Welcome Page</td>
<td>123</td>
</tr>
<tr>
<td>117</td>
<td><a href="http://localhost:8082/partnerapp">http://localhost:8082/partnerapp</a></td>
<td>124</td>
</tr>
<tr>
<td>118</td>
<td>Loading Page 1</td>
<td>124</td>
</tr>
<tr>
<td>119</td>
<td>Loading Page 2</td>
<td>124</td>
</tr>
<tr>
<td>120</td>
<td>Login with SSO</td>
<td>125</td>
</tr>
<tr>
<td>121</td>
<td>User Information</td>
<td>125</td>
</tr>
<tr>
<td>122</td>
<td>Cloud</td>
<td>126</td>
</tr>
<tr>
<td>123</td>
<td>Create Identity Appliance for Cloud Example</td>
<td>127</td>
</tr>
<tr>
<td>124</td>
<td>Conform Identity Appliance for Cloud</td>
<td>127</td>
</tr>
<tr>
<td>125</td>
<td>Create Identity Provider for Cloud</td>
<td>128</td>
</tr>
<tr>
<td>126</td>
<td>Identity Provider Contact tab for Cloud</td>
<td>129</td>
</tr>
<tr>
<td>127</td>
<td>Create Identity Vault for Cloud</td>
<td>130</td>
</tr>
<tr>
<td>128</td>
<td>Create Salesforce Entity</td>
<td>130</td>
</tr>
<tr>
<td>129</td>
<td>Create Google app Entity</td>
<td>131</td>
</tr>
<tr>
<td>130</td>
<td>Create the user <a href="mailto:josso@cloudjosso.com">josso@cloudjosso.com</a></td>
<td>131</td>
</tr>
<tr>
<td>131</td>
<td>Start Running the Identity Appliance</td>
<td>132</td>
</tr>
<tr>
<td>132</td>
<td>Certification</td>
<td>132</td>
</tr>
<tr>
<td>133</td>
<td>Login in Google app Application</td>
<td>134</td>
</tr>
</tbody>
</table>
Picture.134 Advanced Tools Menu ................................................................. 134
Picture.135 Set up Single Sign-On ................................................................. 135
Picture.136 Metadata .................................................................................... 135
Picture.137 Sign-out Page URL ................................................................. 136
Picture.138 Set up Sign-out ................................................................. 136
Picture.139 Save Changes Button ................................................................. 137
Picture.140 Login in Salesforce Application .................................................. 138
Picture.141 Setup Salesforce ................................................................. 138
Picture.142 Security Controls on Salesforce .................................................. 139
Picture.143 Single Sign-On Setting Option on Salesforce .............................. 140
Picture.144 Edit Button ................................................................................ 140
Picture.145 SAML Enabled Button .............................................................. 141
Picture.146 SAML Version ........................................................................... 141
Picture.147 Identity Provider Certification ..................................................... 142
Picture.148 Single Sign On Salesforce .......................................................... 143
Picture.149 URL ........................................................................................... 144
Picture.150 Login Page ................................................................................ 145
Picture.151 JOSSO Processing ................................................................. 145
Picture.152 Login on Salesforce ................................................................. 146
Picture.153 Login in Google apps ............................................................... 146
Picture.154 Automatically Login ................................................................. 147
Picture.155 Login on Cloud mail ............................................................... 147
Picture.156 Gmail ....................................................................................... 148
Picture.157 WiKID ....................................................................................... 150
Picture.158 WiKID Login ................................................................. 150
Picture.159 Configure WiKID ................................................................. 151
Picture.160 Create an Intermediate CA ....................................................... 151
Picture.161 CA fields ................................................................................... 152
Picture.162 Generate Button ....................................................................... 152
Picture.163 Submit for Processing Button ................................................ 153
Picture.164 Certificate ................................................................................ 153
Picture.165 Install Certificate ..................................................................... 154
Picture.166 Install Intermediate Certificate Button .................................... 154
Picture.167 Create a localhost Certificate .................................................. 155
Single Sign-On

Picture.168 localhost Certificate Fields .................................................. 155
Picture.169 Success Window ........................................................................ 156
Picture.170 Enable Network Protocols ......................................................... 156
Picture.171 Network Client ........................................................................ 157
Picture.172 Create a Domain ................................................................. 157
Picture.173 WiKID token .......................................................................... 158
Picture.174 Registration .......................................................................... 158
Picture.175 Add WiKID in JOSSO ........................................................ 159
Picture.176 WiKID Definition Window ...................................................... 160
Picture.177 Connect WiKID with JOSSO ................................................ 160
Picture.178 Login with Passcode ............................................................. 161
Picture.179 Get Passcode ......................................................................... 161
Picture.180 User Login ........................................................................... 162
Abstract

The purpose of the present assignment is to describe JOSSO which is a Java-Based SSO solution for web applications. The scope of this assignment is to give an overall knowledge of JOSSO. More particularly, there will be described the installation steps for JOSSO C.E. and JOSSO E.E. Also it will be analyzed the entire screen interface that JOSSO has. Finally it will be described, through steps, an example of how JOSSO creates simple Single Sign-On environment. Furthermore it will be described an example of a Single Sign-On solution in combination with Cloud computing and finally it will be given an example with a two factor authentication solution with the use of WiKID Strong Authentication System. It would be gratifying that this paper might be useful to anyone that wants to use JOSSO as Single Sign-On solution.

The author,
Tzani Adamantia
1. Introduction

Nowadays, where the common web users’ needs for web services are rising in a high rate day by day, and he is obliged to keep a number of different identities for a corresponding number of applications, Single Sign-On is the solution for this redundancy. When SSO is introduced into systems user experience could be much better. User does not have to remember many different logins and passwords. He needs to remember only one identity and has to login once in order to access all his preferred web applications. There are many SSO solutions available, others commercial and others open source. One of the most prominent is JOSSO. It is an open source Internet SSO solution for standardized Internet-scale SSO implementations, allowing secure Internet access to the Web-based applications or services of customers, suppliers, and business partners. It is one of most preferred SSO options due to its easy deployment that does not require proficient technological expertise.

2. What is JOSSO

JOSSO, also mentioned as Java Open Single Sign-On, is a SSO solution for web applications. It is an open source software based on Java EE for user authentication and authorization. It is released under the GNU Lesser General Public License (LGPL). With the Single Sign-On property of JOSSO a user logs in once and gains access to all systems without being prompted to log in again at each of them.

A deployment of a Single Sign-On solution often means the waste of significant resources for enabling the Single Sign-On applications and setting up the providers of identity that grand user authorization. JOSSO solves that drawback. By using the agent architecture we can apply these capabilities in a transparent mode. Another feature is that it can support third-party applications that are not open source in order to enable a Single Sign-On solution.

More specifically, JOSSO support the following features:

- **Federated Single Sign-On**: Establishment of inbound Identity Provider and outbound Service Provider.
- **SAML Support**: JOSSO is designed to support SAML 2.0, a standard for exchanging authentication and authorization data between security domains.
- **Bundled with Graphical User Console**: The Atricore Console provides a complete visual setup in order to facilitate the user interaction.
- **Transparent**: It supports an immediate Single Sign-On establishment for applications like Java EE, Windows.NET, PHP, Perl, and others.
- **Wide Application Server Support**: It can run under web containers such as JBoss, Apache Tomcat, Weblogic, Windows IIS and others.
- **Standards-based:** It is based on the SAML, WS-Trust, SPML, Java EE and OSGi standards.
- **Pluggable Framework:** It provides an infrastructure that allows the support of already existing authentication mechanisms and identity repositories.
- **Seamless Integration with Spring Security:** It collaborates with Spring Security, which is a Java EE framework that provides authentication, authorization and other security features for enterprise applications.
- **Multiple Authentication Mechanisms:** It uses client certificate authentication, Windows authentication, LDAP authentication and others in order to support identity establishment.
- **Support for Self-Services:** It supports password recovery functionality.

Another important thing that we should mention is that JOSSO is running on Apache Karaf which is a small OSGi based runtime that provide a lightweight container onto which various components and applications can be deployed. The features that Apache Karaf has are:

- **Hot deployment:** Karaf supports hot deployment of OSGi bundles by monitoring jar files inside the [home]/deploy directory. Each time a jar is copied in this folder, it will be installed inside the runtime.

- **Dynamic configuration:** Its services are usually configured through the ConfigurationAdmin OSGi service. Such configuration can be defined in Karaf using property files inside the [home]/etc directory.

- **Logging System:** using a centralized logging back end supported by Log4J, Karaf supports a number of different APIs (JDK 1.4, JCL, SLF4J, Avalon, Tomcat, OSGi)

- **Provisioning:** The provisioning of libraries and applications is done through a number of different ways, by which they will be downloaded locally, installed and executed.

- **Native OS integration:** Karaf can be integrated into our own Operating System as a service so that the lifecycle will be bound to our Operating System.

- **Extensible Shell console:** Karaf features a text console where we can manage the services, install new applications or libraries and manage their state.

- **Remote access:** We can use any SSH client to connect to Karaf and issue commands in the console.

- **Security framework:** It is based on JAAS which is the Java implementation of the standard Pluggable Authentication Module (PAM) information security framework.
Managing instances: Karaf provides simple commands for managing multiple instances. We can create, delete, start and stop instances of Karaf through the console.

3. JOSSO Editions

JOSSO comes in two different editions. The first is the Community Edition (CE) and the second the Enterprise Edition (EE). The JOSSO CE main feature is that it is frequently updated with new versions and comes with LPGL open source license. On the other hand, JOSSO EE is designed for enterprise use, which is fully supported. It comes with a subscription and support package that permits enterprises to configure their own Single Sign-On settings over an extended period. The support period for JOSSO EE is four years.

More specifically JOSSO CE has the following features:

- It is free.
- It supports unlimited number of users.
- It has only one node thought it cannot run in cluster.
- It is designed to run on open source platforms.
- It provides an SAML connection setup for linking SSO and Cloud Providers, but it refers to technically advanced users.

On the contrary JOSSO EE has the following features:

- For small organizations it supports maximum 2000 users with two nodes and for big organizations it supports maximum 10000 users with four nodes.
- It runs on open source platforms but also commercial platforms like Oracle, Windows IIS, MS SQL Server and others.
- It provides scalability, high availability and clustering.
- It provides system monitoring through advanced JMX tools.
- It offers cloud-ready multi-tenant architecture.
- It is set up for Windows Single Sign-On.
- The manufacturer provides technical support through Service Level Agreements (SLAs) along with stress testing.
- It supports embedded cloud SaaS services like Google Apps, Saleseforce and others.
- It also provides automatic upgrades and maintenance, like Service Packs and updates.
4. JOSSO Versions

JOSSO version 1 is the first application that was created. Due to its long period of usage and development it is a stable and reliable solution for transparent SSO focused in introducing End-to-End SSO features.

The transparency capability is achieved due to the compliance with standards such as the ones that Java EE platform offers. This feature enables the applications which rely on the platforms’ security contracts to execute Single Sign-On operations with no integration effort.

Because JOSSO supports a big variety of application vendors it is an attractive solution to interconnect applications designed for different platforms. Furthermore, it is extensible in a way that offers variability within the access management layer.

One drawback of JOSSO1 is that it cannot interact with third-party Single Sign-On applications that exist on external domains. For example, there is no support in providing SaaS provider its security context.

From the usability point of view, the setup process requires technologically high advanced users that are able to handle configuration descriptors and to understand SSO components along with the basic infrastructure.

Subsequently, JOSSO1 is extensible in a way to support mainstream application platforms, authentication mechanisms and identity stores, but its SSO protocol is compatible only with the application and cannot incorporate SAML or OpenID functionalities. Another disadvantage is that it cannot operate in order to offer Multitenancy.

Furthermore, is designed only for SSO services leaving accounting, entitlement management and storage to third-party software, resulting in bigger efforts and investment in order to fill in the gaps.

JOSSO2 is the next generation of the JOSSO products. It is an all-in-one solution providing end-to-end distribution of Federated Single Sign-On settings, based on a model-driven approach, enabling the limited time-to-value. It supports accounting, entitlement management and storage on an internal RDBMS store.

It can deliver Single Sign-On solutions on an Internet scale by communicating and collaborating with other external or cloud providers that have their own Single Sign-On backbone infrastructure.
In case that the setup scenario is based on custom user settings, for example the implementation of a custom Single Sign-On plug-in that supports identity sources or authentication mechanisms that are not common, first generation JOSSO might be more appropriate solution.

For more usability and productivity, JOSSO utilizes the Atricore Console Rich Internet Application or RIA. The user is working in an architecture level by designing own Single Sign-On settings.

5. Provision and analysis of High-availability and Scalability

In JOSSO EE exist two major high-availability elements. These are the System Failover and the Session Failover. With the combination of these two key elements it is guaranteed that does not exist any point of failure in the deployment and that JOSSO EE applications is available anytime.

5.1 System Failover

System failures are common in computer systems. The main reasons are hardware, process or load balancer malfunctions. The recommended action is that there must be installed an adequate number of JOSSO EE servers along with load balancers in order to act as backups or to fail over to. This action guarantees that there are no single points of failure that can compromise the proper function of the application.

5.2 Session Failover

This feature ensures that the session data are accessible from the JOSSO EE servers. In order to deter data loss, the service requests are routed to an alternative failover server. This enables JOSSO EE Service to maintain the sessions into an authenticated state and continue to process new user’s requests that follow the failure. If this feature did not exist, the user should re-authenticate. The session failover has less significance in the case where users need to have read but not write data.

5.3 Failover Mechanism Architecture

In the following diagram it is visible the required elements for the system and session failover feature in JOSSO EE deployment, and these elements are the following:

- A number of JOSSO SSO agents that serve as backups in case of a system failure.
- A load balancer that delivers the workload to the JOSSO EE agents.
- A number of JOSSO EE servers with an embedded state cache that act as system backups in case of system failure. The embedded state cache feature ensures that the replicated state is present during a system failure.
- A number of load balancers that can deliver the workload between the JOSSO EE servers.
- If the JOSSO EE is adjusted for session failover, the configuration and session data are replicated between the servers. In case of a system failure the replicated data are always available.
- A number of Apache Derby DBs in order to store configuration data. These DBs are adjusted for configuration failover.

In the following example the load balancers represent the only access points to JOSSO EE servers.

*Picture 1 Failover Mechanism Architecture*
5.4 Enabling Clustering

The JOSSO EE needs its OSGi bundles to be active in order to support clustering. This is achieved by replacing the “featureBoot” property in the following file `org.apache.karaf.features.cfg` in the `$JOSSO_HOME/etc/` folder, with the following string:

```
featuresBoot=atricore-branding,config,ssh,management,spring,spring-dm,josso-ee-ha
```

The difference is that it is activated the josso-ee-ha feature instead of the josso-ee.

Next, the XML descriptor `$JOSSO2_HOME/etc/ehcache-ha.xml` adds the following data:

```
<cacheManagerPeerProviderFactory
   class="net.sf.ehcache.distribution.RMICacheManagerPeerProviderFactory"
   properties="hostName=node1.acme.com,peerDiscovery=automatic,
                multicastGroupAddress=230.0.0.1,
                multicastGroupPort=4446, timeToLive=32"
   propertySeparator=","
/>`
```

5.5 Configuration Replication

The implementation of clustering support enables the functionality of state replication, but it does not encompass the replicated configuration data. There are three basic ways in order to approach the synchronization of configuration data, the manual, the managed automatic and the non-managed automatic configurations.

**Manual Configuration Replication:** This feature enables the user to control the Identity Appliance Import and Export functionalities keeping them synchronized.

**Managed Automatic Configuration Replication:** This feature is based on the Apache Derby’s failover capabilities. These are:

- One master, one slave.
- Roll-forward shipped log.
- Asymmetry.
- Asynchronicity.
- Shared nothing.
- Replication granularity.

In order to enable data failover using the Apache Derby replication for a JOSSO EE server it must be referred to the descriptor located at `$JOSSO2_HOME/etc/` folder in the `com.atricore.idbus.console.db.cfg` file.
Non-Managed Automatic Configuration Replication: The third option must be considered when it is needed a highly available database to store configuration data. To achieve that, the high availability features are moved from the JOSSO EE server to the external repository. In order to do that the file `com.atricore.idbus.console.db.cfg` placed in the `$JOSSO_HOME/etc/` folder must be updated with the details of the used JDBC driver.


The communication channels on which an Identity Appliance can provide network services are the browser-facing and application-facing channels. The existence of browser-facing channels helps the message exchange between users behind some web browser. An example is when a user submits his credentials to an Identity Provider or when a SAML2 authentication is initiated on a Service Provider. On the other hand, the application-facing communication channels are serving as means for message exchange between application-based users, conducting Application-to-Application communication. The role of external consumers that utilize application-facing channels is played by the JOSSO Agents. For example, the usage of SOAP service innovation helps the user details to pass from the JOSSO Agents to a server-side service that handles requests on an application-facing channel. Continuing, there are followed certain steps in order to enable SSL support for JOSSO2.

6.1 Creating a server certificate with keytool

Using the command below, a key pair and a certificate are generated into the Java keystore by the usage of keytool. The command can be run from the command line.

```
keytool -keystore jetty.keystore -alias jetty -genkey -keyalg RSA
Enter keystore password: secret
What is your first and last name?
[Unknown]: sso.acme.com
What is the name of your organizational unit?
[Unknown]: it
What is the name of your organization?
[Unknown]: acme
What is the name of your City or Locality?
[Unknown]:
What is the name of your State or Province?
[Unknown]:
What is the two-letter country code for this unit?
[Unknown]:
Is CN=sso.acme.com, OU=it, O=acme,
L=Unknown, ST=Unknown, C=Unknown correct?
```
Enter key password for <jetty>
(RETURN if same as keystore password): secret1

Then the generated keystore file named jetty.keystore is copied in the folder named $JOSSO2_HOME/etc. It must be noted that in order for JOSSO2’s web server to get the server certificate, the key alias must be in “jetty” format.

### 6.2 Enabling SSL in the Web Server used by JOSSO

Secure Sockets Layer (SSL), is a cryptographic protocol that provide communication security over the Internet. SSL encrypt the segments of network connections at the Application Layer for the Transport Layer, using asymmetric cryptography for key exchange, symmetric encryption for confidentiality, and message authentication codes for message integrity.

Here the user must edit the org.ops.pax.web.cfg file located in $JOSSO2_HOME/etc, uncommenting the following lines:

```ini
.ops4j.pax.web.ssl.keystore=/opt/atricore/josso-ee-2.2.2/etc/jetty.keystore
org.ops4j.pax.web.ssl.password=secret
org.ops4j.pax.web.ssl.keypassword=secret1
org.osgi.service.http.port.secure=8443
org.ops4j.pax.web.ssl.clientauthwanted=false
org.ops4j.pax.web.ssl.clientauthneeded=false
```

### 6.3 Exporting Web Server SSL Certificate

With this command, can be extracted a JOSSO Web module server certificate, which is in DER format:

```bash
keytool -exportcert -alias jetty -keystore jetty.keystore -file josso-der.crt
```

The next action is to convert it into a PEM format, which is the most common file used by applications like OpenSSL in order to make the PKCS12 conversion.

```bash
openssl x509 -out josso-pem.crt -outform pem -in josso-der.crt -inform der
```

This will result a certificate that can be used by agents in order to access through HTTPS the JOSSO2 web services.
6.4 Agents connection to Web Services over SSL in JOSSO

In the case where the usage of HTTPs protocol in the Identity Appliance is needed the JOSSO Web Server certificate must be marked as trusted, unless the communication is made through a reverse proxy and in that case the proxy certificate must be marked as trusted.

During the Identity Appliance set up, it is possible that the generated agent configuration file could be edited in order to include SSL information. In the following example, form the Java Agents side; it is necessary that the server certificate must be added to the trusted certificates keystore.

# Set to 'On' to enable secure SOAP clients with HTTPS/SSL
GatewayEndpointSSLEnable : On
# Path to server certificate file that stores trusted certificates (needed to verify server)
SSLServerCertFile : /opt/atricore/josso-ee-2.2.2/etc/josso.pem.crt
7. High Level Architecture of JOSSO
Steps:

1a) The user through its browser requests a web service from the Service provider Execution Environment.

1b) In the same time the browser sends an authentication request through the HTTP method GET-POST to the JOSSO Web Authentication Front of the JOSSO Gateway, through Auth Front-Channel Interface.

The Service Provider Execution Environment consists of the following components:

- JOSSO Agent
  - Adapter
  - Authentication & Authorization
  - Identity Services Stubs
  - Web Services Stack
- Security APIs
- Application

Internally in the Service Provider Execution Environment we have the following steps:

1a.1) The Adapter sub-component of JOSSO Agent component uses the JOSSO Agent configuration and Security APIs in order to provide the Security Context that is consumed by the Application.

1a.2) Then the Application requests Authentication & Authorization from the Authentication & Authorization component.

1a.3) If the previous step is executed then the Identity Services Stubs component is executed. Identity Services Stubs is a program when executed provides linking and joining information about an entry across multiple identity management systems.

1a.4) The previous step, if successful, leads to the Web Services Stack component which is the sum of certain technologies such as the Extensible Markup Language (XML) or the Service-Oriented Access Protocol (SOAP).

1c) In this step the JOSSO Agent component after the establishment of the user and his role, sends his credentials in SOAP/HTTP format through the SSO Identity Management Interface to the SSO Identity Management Service, which is a component of the JOSSO Gateway.

1d) At the same time JOSSO Agent sends an Authentication Assertion of the user to SSO Identity Provider Service component through the SSO Identity Provider Interface.
1e) Next, an access session is sent from the JOSSO Agent to the SSO Session Manager Service component thought the SSO Session Manager Interface.

2) The four components are then integrated into the single component named JOSSO Core. The main sub-component of JOSSO Core is the Authenticator from which they derive the following sub-components that are described in the following steps.

2a) The Authentication Scheme component that leads to the Identity Store component, in which are all the identities of the users.

2b) The Session Manager that leads to the Session Store component which controls each session that a user has requested.

2c) The Assertion Manager component that is used by the Authenticator in order to access the Assertion Store which consists of the aggregate of all the Authentication Assertions (SAML, SOAP, XML) that exist in JOSSO.

2d) The Audit & Event Manager component, that it is used for monitoring information and logging history.

3a) When all the processes within the JOSSO Core component are completed, it sends an Identity Storage Access Protocol to the Identity Storage Resource and finally

3b) a State Storage Access Protocol to the Application State Storage Resource.
8. Activity Diagram (State Machine)

**Picture.3 State Machine**

**Steps:**

1) The requested resources are initially provided in order to be checked if a Service Provider SSO is present.

2a) If it is present, they advance to the asserting SSO session.

2b) If not, an Authentication Request is submitted in order to check if the Identity Provider Session is present.

3a) The same procedure as in the previous step (2a) is executed is the SSO session is invalid.

3b) If the session is valid, the Security Context is injected in the simple state where the resource access request is authenticated.
4a) If the SSO session is present, it is created an assertion for the existing SSO session, in order to be relied.

4b) If the SSO session is not present the credentials are requested.

4b1) If these credentials are valid a SSO session is created in order to be relied.

4b2) In the case that they are invalid they return to the previous state for checking. If despite the repeated checks are still in the invalid state, after a maximum number of retries is reached, the program ends.

5) If the assertion is relied is sent to the Service Provider where this request is pending.

6) Then the assertion request is resolved.

7) After that step, the resource accesses and associates with a SSO token to the Service Provider session in order to inject the security context and to be authenticated.

8) In the next step the resource is requested.

9) After the provision of the resource the program exits.

9. How to install JOSSO2
Prior to the description of the installation process of JOSSO2, it is imperative to refer to its directory structure as it is shown below:

-<JOSSO_HOME>/ - the path of our JOSSO installation.
  |-- appliances
  |-- bin
  |-- data
  | |-- cache
  | |-- derby
  | |-- generated-bundles
  | |-- log
  | |-- port
  | |-- work
  |   | |-- repository
  |   | |-- config
  |-- lib
  |-- lock
  |-- system
The appliances folder contains identity appliances managed by JOSSO. The layout of its content follows the same one used for Apache Maven repositories.

The bin folder contains the starting scripts.

The data folder contains files that inform about the application state, like the OSGi bundle cache, database files, log files and temporary artifacts as a result of transformation procedures.

The lib folder contains the libraries for the OSGi Microkernel Implementation on top of which JOSSO builds.

The lock folder is used to support fail-over settings when more than one JOSSO instance is used.

The system folder contains the libraries that determine the JOSSO distribution. Its layout is based on the Apache Maven repository.

10. Installation steps for JOSSO2

First we should mention that we will install JOSSO2 in Ubuntu 11.04 64bit OS. JOSSO2 is compatible also with Microsoft Windows OSs. The prerequisites are JDK 6 or later version and JOSSO2 Community Edition or Enterprise Edition which is a .jar file.

We download JDK and josso-ce-2.2.2-unix.jar respectively. Finally as JOSSO2 Community and Enterprise Edition incorporates the Atricore Console Rich Internet Application (RIA) we need to install in our computer the Adobe Flash player in order to view the Atricore Console.

When we have all the prerequisites we create a folder with the name JOSSO2_HOME. The reason why we choose this name for the folder is because, as the JOSSO2 is running, the Karaf server “sees” that home folder. The next step that we have to do is to run the josso-ce-2.2.2-unix.jar file with JDK 6 and then we have to follow the instruction windows as it is illustrated in pictures 9 to 15. In these instruction windows we will choose to install JOSSO into the JOSSO2_HOME folder. The difference in installation between JOSSO CE and JOSSO EE is that in JOSSO EE we have to agree with the license terms and to upload the Activation license file.

Furthermore, after the installation of JOSSO we open a command line console and type the following commands:
1. `JAVA_HOME=/usr/lib/jvm/java-java-6-openjdk-amd64/`
2. `export JAVA_HOME`

![Installation of JOSSO](image)

*Picture 4 Install JOSSO*

We type the above commands because without them the Java CLASSPATH will not be updated and `java` or `javac` commands will not be recognized.

Then we go to the bin folder, which is into the JOSSO2_HOME folder and type the command `/atricore`. With this command JOSSO2 starts running. Before we type this command we have to choose the Atricore executable to run as a program, by right clicking it and selecting the “Run as a program” checkbox from the properties tab.
Furthermore we can use the following command to determine that all JOSSO modules are up and running: `osgi:list | grep Atricore` as illustrated in the following picture.
As JOSSO2 Community Edition is running, we open a browser (Firefox) and navigate to the Atricore page. Atricore is running in 8081 port. To open the Atricore page we type on the browser `localhost:8081/atricore-console/index.jsp`
Then we login to the Atricore Console with **Username:** admin and **Password:** admin and the Atricore console window is loaded.
The following windows illustrate the installation of JOSSO EE.

*Picture 8 Login Page*

*Picture 9 Step1 of JOSSO EE Installation*
Picture.10 Step 2 of JOSSO EE Installation

Picture.11 Step 3 of JOSSO EE Installation
Picture.12 Step 4 of JOSSO EE Installation

Picture.13 Step 5 of JOSSO EE Installation
Step 6 of JOSSO EE Installation

1. Start JOSSO Enterprise Edition by running the "atricore" command located within the "bin" directory in the target path.

2. You'll be ready to run the Atricore Web Console. In order to launch it, hit the following URL: http://jossoserver:8081/atricore-console
   You should be able to access the server directly from your local system (localhost) as follows by using this URL: http://localhost:8081/atricore-console

3. The product activation dialog will appear. In order to activate JOSSO Enterprise Edition, enter your license key.

4. Finally, sign in using the default credentials: 'admin' as the username and 'admin' as the password.

Use the following to help you get started with JOSSO Enterprise Edition:

Step 7 of JOSSO EE Installation

Installation has completed successfully.
An uninstaller program has been created in:
/home/two/JOSSO-ee/Uninstaller

Generate an automatic installation script
After we finished the installation, we open a browser (Firefox) and navigate to the Atricore page in the same way as in JOSSO CE. The only difference is that the first time that we will try to login we have to browse and find the activation file, as it is illustrated in the following picture.

![Activation License Window](image)

*Picture 16 Activation License Window*

After that step, we press the log in button the Atricore Console loads and we see the Identity Appliance Modeler screen as illustrated in the following picture.
At the beginning we are logged in the Atricore Console as an administrator. Being an administrator, we also have the ability to change the password.

By clicking on “Change Password” option it appears the “Change Password” window as shown in the following picture:
In this window we can enter the password that we have in “Original Password” field, and then enter the new Password in “New Password” field and reenter the new Password in “Confirm Password” field in order to confirm that the new Password is correct. As we have filled in the entire field correctly we click on “Confirm” button in order to apply the changes or the “Cancel” button if we want to abort changes.

Moving on, it is time to describe the Atricore Console as it is illustrated in Picture: 17

As we observe in Picture: 20 we see that it consists of a bar with five options

- Identity Appliance Modeler
- Identity Appliance Lifecycle Management
- In Account & Entitlement Management
- System Setting
- Help

Firstly we will describe the Identity Appliance Modeler. In the Identity Appliance Modeler window we draw our Identity Appliance. This console is consists of the:

1. Action Bar
2. Diagram Canvas
3. Pallet
4. Property Sheed
5. Browser

The Action Bar has buttons like open, new, save that gives us the opportunity to open an existing Identity Appliance or to create a new one or to save our work. In the Diagram Canvas we will draw our model using the principal entities from the Pallet. In Property Sheed we can see the detail of each principal entity that we have in our model in the Diagram Canvas. Finally in Browser we can see the Identity Appliance with more details in a form of a tree diagram.

One of the most important parts of JOSSO2 is the Identity Appliance Modeler screen in which we draw our Identity model.

In the following section we are going to describe first what an Identity Appliance is and then we are going to describe the principals and the entities that we can use to create the Identity Appliance.

11. Identity Appliance

Internet SSO solutions are known within JOSSO as "Identity Appliances". An Identity Appliance is a component which encloses the necessary definitions that instantiate the SSO services, in order to apply particular identity architecture.

For example, during the deployment of an Identity Appliance, predefined SSO endpoints are enabled. Each endpoint displays a specific behavior for a particular authentication service or identity data streams from an arbitrary identity store corresponding to a specific user schema.

The Identity Appliances can be defined either by using the Atricore Console or directly, using a textual notation based on XML descriptors.

Identity Appliancees are standard OSGi Bundles, as are all the artifacts that make up JOSSO. A bundle is a file that contains all the information about a given component. This file is in "jar" format, resembling the "jar archives" that Java developers use to distribute libraries of code. The bundle is composed of a MANIFEST.MF file which specifies its identity and the bundles upon which it depends, as well as what is visible for other bundles to use.
Apache Maven is the preferred system for packaging an Identity Appliance. Apache Maven is a build automation tool typically used for Java projects. It is based on the concept of a project object model (POM).

Moreover, the Identity Appliances generated by the Atricore Console include an incorporated descriptor named, pom.xml, for packaging the Identity Appliance in an out-of-the-box way with the usage of Apache Maven version 3. Furthermore, the layout used only by JOSSO2 to define Identity Appliances, is that used by Apache Maven.

A Maven repository is used to hold build artifacts and dependencies of varying types. There are strictly only two types of repositories: local and remote. The local repository refers to a copy on your own installation that is a cache of the remote downloads, and also contains the temporary build artifacts that you have not yet released. Remote repositories refer to any other type of repository, accessed by a variety of protocols such as file:// and http://. These repositories might be a truly remote repository set up by a third party to provide their artifacts for downloading (for example, repo.maven.apache.org and uk.maven.org house Maven's central repository). Other "remote" repositories may be internal repositories set up on a file or HTTP server within an organization, used to share private artifacts between development teams and for releases.

JOSSO2 hosts two internal Maven Repositories which are accessed when an OSGi bundle is to be installed.

The first is located in $JOSSO_HOME/system folder that contains the executable artifacts JOSSO2. Here we can find also both JOSSO2 artifacts and their dependencies.

The second is located in $JOSSO_HOME/appliances folder that contains the Identity Appliances. Each time an Identity Appliance is generated and packaged using the lifecycle management provided in the Atricore Console, it is dropped into this folder location.

The feature descriptors are utilized to simplify the provisioning of capabilities within JOSSO2. This mechanism is provided by a set of commands available in the features shell. JOSSO2 features are stored in the $JOSSO_HOME/features directory which complies with the Maven repository layout.

For example, the main features descriptor for the JOSSO2 distribution looks like this, as we can see below:
<?xml version="1.0" encoding="UTF-8"?>
<features name="atricore-josso-ce-2.2.0">
  <!-- JOSSO CE -->
  <feature name="josso-ce" version="2.2.0">
    <feature version="1.1.0">common</feature>
    <feature version="2.2.0">josso-ce-svcs</feature>
    <feature version="1.1.0">atricore</feature>
    <feature version="1.1.0">atricore-management</feature>
    <feature version="2.2.0">josso-ce-console</feature>
  </feature>
  <!-- JOSSO CE Servies -->
  <feature name="josso-ce-svcs" version="2.2.0">
    <!-- License Manager service -->
    <bundle start-level="35">mvn:com.atricore.idbus.console.licensing/com.atricore.idbus.console.licensing.josso2-license-v1_0/1.0.0</bundle>
    <bundle start-level="35">mvn:com.atricore.idbus.console.licensing/com.atricore.idbus.console.licensing.main/1.0.0</bundle>
    <bundle start-level="35">mvn:com.atricore.idbus.console.licensing/com.atricore.idbus.console.licensing.command/1.0.0</bundle>
  </feature>
  <!-- JOSSO CE Console -->
  <feature name="josso-ce-console" version="2.2.0">
    <feature version="2.2.0">josso-ce-console-svcs</feature>
    <feature version="2.2.0">josso-ce-console-web</feature>
  </feature>
  <!-- JOSSO CE Console Services -->
  <feature name="josso-ce-console-svcs" version="2.2.0">
    <bundle>mvn:com.atricore.idbus.console.appliance/com.atricore.idbus.console.appliance.console-default-idau/1.0.0</bundle>
    <bundle>mvn:com.atricore.idbus.console.appliance/com.atricore.idbus.console.appliance.jaas/1.0.0</bundle>
    <bundle>mvn:com.atricore.idbus.console.activation/com.atricore.idbus.console.activation.protocol/1.0.0</bundle>
    <bundle>mvn:com.atricore.idbus.console.activation/com.atricore.idbus.console.activation.main/1.0.0</bundle>
    <bundle>mvn:com.atricore.idbus.console.activation/com.atricore.idbus.console.activation.command/1.0.0</bundle>
  </feature>
</features>
As we have described what is the Identity Appliance we will continue with the description of the principals and the entities that we can use from the “Pallet” to create our model.

12. Entities

The nature of the providers is twofold. They can be internal or external providers. The internal are hosted locally and are build in JOSSO in order to provide Identity and Access Management (IAM) services. The external are hosted remotely, e.g. on a Cloud provider, and they are bases on third-party solutions, independent from JOSSO. Moreover, because the internal providers are hosted within the user’s organization, their installation, setup and lifecycle can be fully monitored and managed. On the other hand, the external providers can be monitored by establishing Federated SSO connections, with the exception that the user has no right to have access to their detailed specifications, their behavior or the management back-end, due to the fact that they are part of the remote host.

In order to create an Identity Appliance first of all, we have to choose from the “Entities” the Identity Provider and Service Provider.

The Identity Provider is responsible for authentication and authorization and also is responsible to create a relationship of confidence with another Service Provider through a federated connection.
The Identity Provider definition according to OASIS\(^1\) is that it “Creates, maintains, and manages identity information for principals, and provides principal authentication to other Service Providers within a federation.”

The Identity Provider (IdP) entity is essential in order to set up the Single Sign On function. This entity is entrusted to manage the user’s identity, to deliver security tokens (SAML) and to provide authentication services for client applications. The security tokens mentioned above include the user’s identity and claims. These tokens are forwarded to the relying parties of the Identity Providers which are known as Service Providers (SP). These new entities can give authorization and access privileges to a user based on the claims that have received from the Identity Providers.

An Identity Provider can be associated with one or more Service Providers. This means that the Service Providers will depend on the requirements provided by the trusted Identity Provider. Their association will be a federation connection or an identity lookup connection. The difference among those two connections is that the federation identity connection establishes a trust relationship between the Identity Provider and the Service Provider, while the identity lookup is not. The SSO exchange system of a federation connection is based on digital signature. This method ensures message authentication, integrity and non-repudiation. On the other hand an identity lookup connection points to the Identity Provider to a specific identity store that provides user and entitlement information.

### 13. Configuration and installation of the Identity Provider’s Identity Store

JOSSO provides four different options of Identity Stores for the Identity Provider. These can be found in the “Identity Sources” drawer from the Palette, and are the Identity Vault Authoritative Source, the DB Identity Source, the LDAP Identity Source and the XML Identity Source.

If the Identity Vault option is selected, firstly it must be defined for the Identity Appliance. When it is defined, the Identity Lookup connection is used from the “Connections” drawer, in order to connect the Identity Vault with the Identity Provider.

In the case that the LDAP Identity Source is selected, as before, it must be defined for the Identity Appliance. After that step, the Identity Lookup connection is used in order to connect the LDAP Identity Source with the Identity Provider.

\(^1\) [https://www.oasis-open.org/](https://www.oasis-open.org/)
Another option is the usage of the DB Identity Source. As in the two previous components, an RDBMS Identity Source element needs to be defined for the Identity Appliance. In order to connect the RDBMS Identity Source with the Identity Provider the Identity Lookup connection is chosen.

Finally, in the last option, which is the XML Identity Source as described before, needs to be defined. Also at this Identity Source option the connection that is used is the Identity Lookup

**14. Execution Environments**

An execution environment is a service where Service Providers run in order to offer services to users and applications.

An execution environment can be a Web container, an Application Server, a Web portal or an Application Platform. The characteristics of an execution environment are defined in the Identity Appliance and are made the bindings between the Service Provider and itself in order to activate the SSO support that extends to the subjacent applications.

An activation connection is implemented in order to bind the Service Provider with the execution environment.

The role of the Service Providers is to offer applications and services to end-users. In order to accomplish that feature Service Providers are based on third-party software which plays the role of supporting infrastructure considering security, connectivity and communication. For instance, if it is needed to support web applications, there are used web containers\(^2\). The execution environments that are compatible with JOSSO are the Alfresco, the Apache Web Server, the JavaEE, the JBoss Portal, the Liferay Portal, the phpBB, the Web Server, the WebLogic Server, the Websphere Community Edition (WASCE), the Windows IIS, the Apache Tomcat and the JBoss platform.

Alfresco is a Free/Libre enterprise content management\(^3\) system for Microsoft Windows and Unix-like operating systems.

The Apache HTTP Server is the most popular open source, standard, secure, efficient and extensible HTTP server for modern operating systems, including UNIX and Windows NT. An Apache Server can run all types of web applications, written in PHP, Perl, Ruby and Python among many others.

Java EE is Oracle's enterprise Java computing platform. The platform provides an API

---

\(^2\) A Web container (also known as a Servlet container) is the component of a web server that interacts with Java servlets.

\(^3\) Enterprise content management (ECM) is a formalized means of organizing and storing an organization's documents, and other content, that relate to the organization's processes.
and runtime environment for developing and running enterprise software, including network and web services, and other large-scale, multi-tiered, scalable, reliable, and secure network applications.

JBoss Portal provides an open source platform for hosting and serving a portal's Web interface, publishing and managing its content, and customizing its experience. It delivers the benefits of a zero-cost open source license combined with a flexible and scalable underlying platform.

Liferay Portal is a free and open source enterprise portal written in Java and distributed under the GNU Lesser General Public License and proprietary licenses. It is primarily used to power corporate intranets and extranets.

The phpBB is an Internet forum package written in the PHP scripting language. The name phpBB is an abbreviation of PHP Bulletin Board. phpBB is free and open source software. Features of phpBB include support for multiple database engines (PostgreSQL, SQLite, MySQL, Oracle, and Microsoft SQL Server), flat message structure, hierarchical subforums, topic split/merge/lock, user groups, and multiple attachments per post, full-text search, plugins and various notification options.

A web server execution environment represents a generic web server (or container) hosting web applications or resources.

Oracle WebLogic Server is an application server for cloud and conventional environments and consists of a Java EE platform product-family that includes a Java EE application server, an enterprise portal, WebLogic Portal, an Enterprise Application Integration platform, a transaction server and infrastructure, WebLogic Tuxedo, a telecommunication platform, a WebLogic Communication Platform and a HTTP web server.

WebSphere Application Server Community Edition (WASCE) is a free, certified Java EE 6 application server for building and managing Java applications. It is IBM's supported distribution of Apache Geronimo that uses Tomcat for servlet container and Axis 2 for web services.

Internet Information Services (IIS) is a web server application and set of feature extension modules created by Microsoft for use with Microsoft Windows. The 7.5 edition supports HTTP, HTTPS, FTP, FTPS, SMTP and NNTP.

Apache Tomcat is an open source web server and servlet container. Tomcat implements the Java Servlet and the JavaServer Pages (JSP) specifications from Oracle Corporation, and provides a HTTP web server environment for Java code to

---

4 An enterprise portal, also known as an enterprise information portal (EIP) or corporate portal, is a framework for integrating information, people and processes across organizational boundaries.
run. It includes tools for configuration and management, but can also be configured by editing XML configuration files.

JBoss Application Server is an application server that implements the Java Platform, Enterprise Edition (Java EE). It is written in Java and it is usable on any operating system that supports Java.

15. Configuration and installation of the Service Provider’s Execution Environment

As we have already mentioned, there are a number of execution environments that can be used by the Service Provider. These exist in the “Execution Environments” drawer where they can be dragged to a desirable place in the Palette. The connection between the execution environment and the Service Provider is achieved by the usage from the “Connections” drawer the “Activation” connection. Here it will be analyzed the setup dialog window of each environment. The execution environments that JOSSO supports are the following:

15.1 Alfresco Execution Environment

The window that opens when the execution environment is deployed on the canvas is as it is shown in the following picture. The fields that are marked with the red asterisk are mandatory.

![Create Alfresco Execution Environment Dialog Window](Picture21.jpg)
Field description:

Name: Here the name of the execution environment is specified.

Description: Some description for the execution environment.

Target Host: There are two options, “Local” and “Remote”. The “Local” option assumes that the execution environment is in the same host that JOSSO is. The “Remote” option assumes that the execution environment is located in a different host.

Install Home: It is the folder where the artifacts of Alfresco Execution Environment exist.

Remote JOSSO2 URL: This field is only visible when the “Remote” option is selected. It represents the endpoint of the activation of the web service for the remote JOSSO2 application.

Container Type: Here the type of the Web container is chosen where the Alfresco Execution Environment is deployed.

Container Home: There is defined the folder where the Web container exist that Alfresco Execution Environment is executing.

Overwrite Original Setup: If this checkbox is selected the original settings of the execution environment are replaced with new ones.

15.2 Apache Web Server Execution Environment

The window that opens when the execution environment is deployed on the canvas is as it is shown on the following picture. The fields that are marked with the red asterisk are mandatory.

---

<sup>5</sup> Web container (also known as a Servlet container) is the component of a web server that interacts with Java servlets.
Field description:

Name: Here the name of the execution environment is specified.

Description: Some description for the execution environment.

Target Host: There are two options, “Local” and “Remote”. The “Local” option assumes that the execution environment is in the same host that JOSSO is. The “Remote” option assumes that the execution environment is located in a different host.

Install Home: It is the folder where the artifacts of Apache Execution Environment exist.

Remote JOSSO2 URL: This field is only visible when the “Remote” option is selected. It represents the endpoint of the activation of the web service for the remote JOSSO2 application.

Overwrite Original Setup: If this checkbox is selected the original settings of the execution environment are replaced with new ones.

Install Demo Applications: If this checkbox is selected a JOSSO example web application is deployed on the execution environment. If it is needed a verification that the setting of the Single Sign-On is working properly, this checkbox must be selected.
15.3 **Java EE Execution Environment**

The window that opens when the execution environment is deployed on the canvas is as it is shown on the following picture. The fields that are marked with the red asterisk are mandatory.

![Create JavaEE Execution Environment](Image)

**Field description:**

**Name:** Here the name of the execution environment is specified.

**Description:** Some description for the execution environment.

**Target Host:** There are two options, “Local” and “Remote”. The “Local” option assumes that the execution environment is in the same host that JOSSO is. The “Remote” option assumes that the execution environment is located in a different host.

**Install Home:** It is the folder where the artifacts of JavaEE Execution Environment exist.

**Remote JOSSO2 URL:** This field is only visible when the “Remote” option is selected. It represents the endpoint of the activation of the web service for the remote JOSSO2 application.

---

*Picture 23 Java EE Execution Environment*
15.4 JBoss Portal Execution Environment

The window that opens when the execution environment is deployed on the canvas is as it is shown on the following picture. The fields that are marked with the red asterisk are mandatory.

![Create JBoss Portal Execution Environment](image)

**Field description:**

**Name:** Here the name of the execution environment is specified.

**Description:** Some description for the execution environment.

**Target Host:** There are two options, “Local” and “Remote”. The “Local” option assumes that the execution environment is in the same host that JOSSO is. The “Remote” option assumes that the execution environment is located in a different host.

**Install Home:** It is the folder where the artifacts of JBoss Portal Execution Environment exist.

**Remote JOSSO2 URL:** This field is only visible when the “Remote” option is selected. It represents the endpoint of the activation of the web service for the remote JOSSO2 application.

**Overwrite Original Setup:** If this checkbox is selected the original settings of the execution environment are replaced with new ones.
Install Demo Applications: If this checkbox is selected a JOSSO example web application is deployed on the execution environment. If it is needed a verification that the setting of the Single Sign-On is working properly, this checkbox must be selected.

15.5 Liferay Portal Execution Environment

The window that opens when the execution environment is deployed on the canvas is as it is shown on the following picture. The fields that are marked with the red asterisk are mandatory.

![Create Liferay Portal Execution Environment](image)

*Picture.25 Liferay Portal Execution Environment*

**Field description:**

**Name:** Here the name of the execution environment is specified.

**Description:** Some description for the execution environment.

**Target Host:** There are two options, “Local” and “Remote”. The “Local” option assumes that the execution environment is in the same host that JOSSO is. The “Remote” option assumes that the execution environment is located in a different host.

**Install Home:** It is the folder where the artifacts of Liferay Portal Execution Environment exist.
Remote JOSSO2 URL: This field is only visible when the “Remote” option is selected. It represents the endpoint of the activation of the web service for the remote JOSSO2 application.

Container Type: Here the type of the Web container is chosen where the Liferay Portal Execution Environment is deployed.

Container Home: There is defined the folder where the Web container exist that Liferay Portal Execution Environment is executing.

Overwrite Original Setup: If this checkbox is selected the original settings of the execution environment are replaced with new ones.

Install Demo Applications: If this checkbox is selected a JOSSO example web application is deployed on the execution environment. If it is needed a verification that the setting of the Single Sign-On is working properly, this checkbox must be selected.

15.6 PhpBB Execution Environment

The window that opens when the execution environment is deployed on the canvas is as it is shown on the following picture. The fields that are marked with the red asterisk are mandatory.

Picture.26 PhpBB Execution Environment
Field description:

Name: Here the name of the execution environment is specified.

Description: Some description for the execution environment.

Target Host: There are two options, “Local” and “Remote”. The “Local” option assumes that the execution environment is in the same host that JOSSO is. The “Remote” option assumes that the execution environment is located in a different host.

Install Home: It is the folder where the artifacts of phpBB Execution Environment exist.

Remote JOSSO2 URL: This field is only visible when the “Remote” option is selected. It represents the endpoint of the activation of the web service for the remote JOSSO2 application.

Overwrite Original Setup: If this checkbox is selected the original settings of the execution environment are replaced with new ones.

Install Demo Applications: If this checkbox is selected a JOSSO example web application is deployed on the execution environment. If it is needed a verification that the setting of the Single Sign-On is working properly, this checkbox must be selected.

15.7 Webserver Execution Environment

The window that opens when the execution environment is deployed on the canvas is as it is shown on the following picture. The fields that are marked with the red asterisk are mandatory.
Field description:

Name: Here the name of the execution environment is specified.

Description: Some description for the execution environment.

Target Host: There are two options, “Local” and “Remote”. The “Local” option assumes that the execution environment is in the same host that JOSSO is. The “Remote” option assumes that the execution environment is located in a different host.

Install Home: It is the folder where the artifacts of phpBB Execution Environment exist.

Remote JOSSO2 URL: This field is only visible when the “Remote” option is selected. It represents the endpoint of the activation of the web service for the remote JOSSO2 application.

15.8 Oracle Weblogic Execution Environment

The window that opens when the execution environment is deployed on the canvas is as it is shown on the following picture. The fields that are marked with the red asterisk are mandatory.
Field description:

Name: Here the name of the execution environment is specified.

Description: Some description for the execution environment.

Version: Here the version of the execution environment based on the Oracle Weblogic is chosen.

Target Host: There are two options, “Local” and “Remote”. The “Local” option assumes that the execution environment is in the same host that JOSSO is. The “Remote” option assumes that the execution environment is located in a different host.

Install Home: It is the folder where the artifacts of Weblogic Execution Environment exist.

Remote JOSSO2 URL: This field is only visible when the “Remote” option is selected. It represents the endpoint of the activation of the web service for the remote JOSSO2 application.

Overwrite Original Setup: If this checkbox is selected the original settings of the execution environment are replaced with new ones.

Install Demo Applications: If this checkbox is selected a JOSSO example web application is deployed on the execution environment. If it is needed a verification that the setting of the Single Sign-On is working properly, this checkbox must be selected.
15.9 Websphere Community Edition (WASCE) Execution Environment

The window that opens when the execution environment is deployed on the canvas is as it is shown on the following picture. The fields that are marked with the red asterisk are mandatory.

![Create WASCE Execution Environment Window](image)

*Picture.29 Websphere Community Edition (WASCE) Execution Environment*

**Field description:**

**Name:** Here the name of the execution environment is specified.

**Description:** Some description for the execution environment.

**Target Host:** There are two options, “Local” and “Remote”. The “Local” option assumes that the execution environment is in the same host that JOSSO is. The “Remote” option assumes that the execution environment is located in a different host.

**Install Home:** It is the folder where the artifacts of Websphere Execution Environment exist.

**Remote JOSSO2 URL:** This field is only visible when the “Remote” option is selected. It represents the endpoint of the activation of the web service for the remote JOSSO2 application.
Overwrite Original Setup: If this checkbox is selected the original settings of the execution environment are replaced with new ones.

Install Demo Applications: If this checkbox is selected a JOSSO example web application is deployed on the execution environment. If it is needed a verification that the setting of the Single Sign-On is working properly, this checkbox must be selected.

15.10 Windows IIS Execution Environment

The window that opens when the execution environment is deployed on the canvas is as it is shown on the following picture. The fields that are marked with the red asterisk are mandatory.

![Create Windows IIS Execution Environment](image)

**Picture.30 Windows IIS Execution Environment**

**Field description:**

Name: Here the name of the execution environment is specified.

Description: Some description for the execution environment.

Target Host: There are two options, “Local” and “Remote”. The “Local” option assumes that the execution environment is in the same host that JOSSO is. The “Remote” option assumes that the execution environment is located in a different host.
Install Home: It is the folder where the artifacts of Windows IIS Execution Environment exist.

Remote JOSSO2 URL: This field is only visible when the “Remote” option is selected. It represents the endpoint of the activation of the web service for the remote JOSSO2 application.

Overwrite Original Setup: If this checkbox is selected the original settings of the execution environment are replaced with new ones.

Install Demo Applications: If this checkbox is selected a JOSSO example web application is deployed on the execution environment. If it is needed a verification that the setting of the Single Sign-On is working properly, this checkbox must be selected.

15.11 Apache Tomcat Execution Environment

The window that opens when the execution environment is deployed on the canvas is as it is shown on the following picture. The fields that are marked with the red asterisk are mandatory.

![Create Tomcat Execution Environment](Picture.31 Apache Tomcat Execution Environment)

**Field description:**

Name: Here the name of the execution environment is specified.

Description: Some description for the execution environment.

Version: Here the version of the execution environment Apache Tomcat is chosen.
**Target Host**: There are two options, “Local” and “Remote”. The “Local” option assumes that the execution environment is in the same host that JOSSO is. The “Remote” option assumes that the execution environment is located in a different host.

**Install Home**: It is the folder where the artifacts of Apache Tomcat Execution Environment exist. The value for this field should correspond to that of the CATALINA_HOME environment variable.

**Remote JOSSO2 URL**: This field is only visible when the “Remote” option is selected. It represents the endpoint of the activation of the web service for the remote JOSSO2 application.

**Overwrite Original Setup**: If this checkbox is selected the original settings of the execution environment are replaced with new ones.

**Install Demo Applications**: If this checkbox is selected a JOSSO example web application is deployed on the execution environment. If it is needed a verification that the setting of the Single Sign-On is working properly, this checkbox must be selected.

### 15.12 JBoss Execution Environment

The window that opens when the execution environment is deployed on the canvas is as it is shown on the following picture. The fields that are marked with the red asterisk are mandatory.
**Field description:**

**Name:** Here the name of the execution environment is specified.

**Description:** Some description for the execution environment.

**Version:** Here the version of the execution environment JBoss is chosen.

**Target Host:** There are two options, “Local” and “Remote”. The “Local” option assumes that the execution environment is in the same host that JOSSO is. The “Remote” option assumes that the execution environment is located in a different host.

**Install Home:** It is the folder where the artifacts of JBoss Execution Environment exist. The value for this field should correspond to that of the JBOSS_HOME environment variable.

**Remote JOSSO2 URL:** This field is only visible when the “Remote” option is selected. It represents the endpoint of the activation of the web service for the remote JOSSO2 application.

**Overwrite Original Setup:** If this checkbox is selected the original settings of the execution environment are replaced with new ones.

**Install Demo Applications:** If this checkbox is selected a JOSSO example web application is deployed on the execution environment. If it is needed a verification
that the setting of the Single Sign-On is working properly, this checkbox must be selected.

16. **Authentication**

Here we can choose among servers that provide authentication services remotely to users and systems. This feature is available only in JOSSO Enterprise Edition. JOSSO cooperates with three different external Authentication servers. These are WiKID, Windows Domain and JOSSO’s own directory service.

The WiKID Strong Authentication System is a public-key based two-factor authentication system. It is a flexible, extensible, and secure alternative to tokens, certs and passwords. Application & API support exists for Java, ASP, PHP, Ruby, OpenVPN, TACACS+, etc.

Windows domain is a collection of security principals that share a central directory database. The central database is usually maintained by a proprietary Microsoft product or technology, known as Active Directory. Each person who uses computers within a domain receives his own unique account, or user name. This account can then be assigned access to resources within the domain.

JOSSO’s directory-based authentication is built onto the bind operation of the LDAP protocol. It authenticates the client to the server using this operation.

17. **Authentication Setup**

17.1 **Configuration an installation of Directory-based Authentication**

The Directory-Based Authentication is based on the LDAP protocol, authenticating a client to a server. The Identity Provider setup for directory-based authentication authorizes an external directory to verify the user credentials remotely through a LDAP Bind request. If this operation is successful the user will be authenticated. In order to build such a directory the next steps are followed.

From the “Authentication” drawer the “Directory Service” entity is chosen and dragged into the Diagram Canvas. This action opens a window named “New Directory Definition” which has two tabs; the “Base” tab and the “Lookup” tab, as it is shown below.
**Field Description:**

**Name:** Here is entered a preferred name for the Directory-based Authentication entity.

**Description:** A short description for this entity.

**Initial Context Factory:** In this field is entered the default JNDI\(^6\) environment property which is the following string, com.sun.jndi.ldap.LdapCtxFactory.

**Provider URL:** Here is entered the provider’s URL. The default value for this is ldap://localhost:389.

**Perform DN Search:** This checkbox, if selected, determines if the DN of a user entry will be used as an identifier. If it is not selected, the user identifier will be put to the common name attribute.

**LDAP Security Policy:** There are two options, “None” and “RFC Draft”. If “None” is selected the LDAP Security Policy is disabled. If “RFC Draft” is selected the security policy that is followed is based on the specifications contained in the draft RFC named draft-behera-ldap-password-policy-09.

**Security Principal:** Here is entered the security principal for user authentication to a service. The default value is “uid=admin, ou=system”.

---

\(^6\) The Java Naming and Directory Interface (JNDI) is a Java API for a directory service that allows Java software clients to discover and look up data and objects via a name.
Security Credential: Here is entered a password for user authentication to a service.

Retype Credential: The password is retyped.

Security Authentication: There are three options. “None” for anonymous binding, “Simple” for password-based authentication and “Strong” for authentication based on X.509 certificates.

The “Lookup” tab is determined how the user and role entries are retrieved. This window is shown below.

![Authentication Lookup tab](image)

**Picture 34 Authentication Lookup tab**

Field Description:

User DN: Here is entered a DN used as a context for user searches. The default value is “ou=People,dc=my-domain,dc=com”.

Principal UID attribute ID: Here is entered a LDAP attribute name. The default value is “uid”.

Search Scope: There are four strategies used for queering user/role entries in a LDAP directory. There is the “Base” option, where queries are set up within specific boundaries. The other is the “One” option where the LDAP queries are searching the immediate children of the LDAP tree representing user and role DNs. The third is the “Subtree” option, where the whole LDAP subtree directory is queried. The fourth is
the “Children” option, where the LDAP queries search one level below from all the direct children of the major entry.

17.2 Configuration and installation of Integrated Windows Authentication

Integrated Windows Authentication (IWA) is a term associated with Microsoft products that refers to the SPNEGO\(^7\), Kerberos\(^8\), and NTLMSSP\(^9\) authentication protocols with respect to SSPI\(^{10}\) functionality introduced with Microsoft Windows 2000 and included with later Windows NT-based operating systems. The term is used more commonly for the automatically authenticated connections between Microsoft Internet Information Services, Internet Explorer, and other Active Directory aware applications.

The usage of that authentication mechanism enables the user that has an open session to a trusted Windows Domain, to automatically login when he wants to access the resources of a Service Provider. In order to build such a directory the next steps are followed.

From the “Authentication” drawer the “Windows Domain” entity is chosen and dragged into the Diagram Canvas. This action opens a window named “New Windows Domain Definition”, as it is shown below.

---

\(^7\) SPNEGO is a "pseudo mechanism" that is used to negotiate one of a number of possible real mechanisms and used when a client application wants to authenticate to a remote server, but neither end is sure what authentication protocols the other supports.

\(^8\) Kerberos is a computer network authentication protocol which works on the basis of "tickets" to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner.

\(^9\) NTLMSSP (NT LAN Manager Security Support Provider) is a binary messaging protocol used by the Microsoft Security Support Provider Interface (SSPI) to facilitate challenge-response authentication and to negotiate integrity and confidentiality options.

\(^{10}\) Security Support Provider Interface (SSPI) is an API used by Microsoft Windows systems to perform a variety of security-related operations such as authentication.
Field Description:

Name: Here is entered a preferred name for the Windows Domain Authentication entity.

Description: A short description for this entity.

Protocol: There are two options. The default is “Kerberos” protocol, which enables the usage of negotiating tickets exchanged with a Windows domain controller. The other option is the “NTLM v2” protocol, which is using negotiating tickets exchanged with a Windows domain controller.

Windows Domain: Here is entered the unique identifier of the trusted Windows domain controller.

Service Class: Here is set the application protocol, like HTTP, that is used by JOSSO for servicing requests.

Host: Here is entered the name of the computer that hosts the service. The default value is “localhost”.

Port: Here is entered the number of the port used by JOSSO for servicing requests. The default port number that JOSSO listens to is 8081.

Service Name: In this field is entered the name of the Service Provider.
Domain Controller: Here is entered the appropriate hostname of the Windows domain controller.

Configure Kerberos: If this checkbox is selected an automatically generating Kerberos configuration is enabled.

Keytab file: This option selects the keytab file from the local file system. This file is used to authenticate the Identity Provider versus the Windows domain controller.

### 17.3 Configuration and installation Two-Factor Authentication

Two-factor authentication (TFA, T-FA or 2FA) is an approach to authentication which requires the presentation of two or more of the three authentication factors: a knowledge factor “something the user knows”, a possession factor “something the user has”, and an inherence factor “something the user is”. WiKID is such an application and works as follows. A user selects the domain he wishes to use and enters the PIN into his WiKID Two-factor client. The client is encrypted with the WiKID Server's public key - assuring that only that server can decrypt it with its private key. If the server can decrypt the PIN and it is correct and the account is active, it generates the one-time passcode (OTP) and encrypts it with the client's public key. The user then enters his username and the OTP into whatever service they are using, a VPN for instance, which forwards it to the WiKID Server for validation.

In order to build such a directory the next steps are followed.

From the “Authentication” drawer the “WiKID 2FA” entity is chosen and dragged into the Diagram Canvas. This action opens a window named “New WiKID Definition”, as it is shown below.
**Picture 36 WiKID Authentication**

**Field Description:**

Name: Here is entered a preferred name for the WiKID Two-Factor Authentication entity.

Description: A short description for this entity.

Server Host: Here is entered the IP address or host name of the WiKID server.

Server Code: Here is entered the server code of the WiKID domain.

Certificate Authority Store: If it is selected, it permits the uploading of the Certificate Authority Keystore of the WiKID server. This keystore file is used for decryption and authentication purposes.

Certificate Authority Password: A password for the Certificate Authority Store.

WiKID Client Store: It uploads the keystore of JOSSO client server, which is used for authenticating and encrypting requests to the WiKID server, to the users’ local system.

WiKID Client Password: A password for the WiKID Client keystore.
18. Identity Sources

The Identity Sources are the data layer that providers are based. This layer is required by identity and access management in order to support authentication and SSO.

Furthermore, the Service Provider can be based on an Identity Source in order to connect the Identity Provider’s account to a local counterpart and utilize the available user details inside the Identity Provider to increase output claims. The association between a Service Provider and an Identity Source is not obligatory. It is made only when argument claims are forwarded from the trusted Identity Providers with available information for a Service Provider local store. In order to define a local Identity Store for a Service Provider, it is used the “identity lookup” connection from the Connections drawer of the Palette. The Identity Sources for the Service Providers are the same as for the Identity Provider and are the Identity Vault, the LDAP Directory, the RDBMS identity source and the XML identity source.

The Identity Vault is a type of Identity Source which is built into JOSSO. This feature is based on the Apache Derby relational database. This type of Identity source can be linked to an Identity Provider in order to support authentication processes and to a Service Provider in order to support account linkage and Identity Mapping.

The LDAP Identity Source can be accessed through the LDAP protocol, which displays the user entries hierarchically. The LDAP Identity Source can be linked to an Identity Provider in order to imply queries that retrieve user records supplied to authentication and related processes that use the LDAP protocol. It can also link to a Service Provider in order to imply queries that locally authenticate users or to boost Identity Provider claims by a directory accessible through the LDAP protocol.

The RDBMS Identity Source can be accessed by the vendor’s Java Database Connectivity driver (JDBC). This driver conceals the internal details of the protocol that it is used to access databases. The user’s credentials and other details are stored in tables in the form of rows. The RDBMS Identity Source can be linked to an Identity Provider for executing queries in order to retrieve user records using the supplied JDBC driver. It can also be linked to Service Provider executing queries for user local authentication or boosting Identity Provider claims thought the JDBC driver.

The XML Identity Source is depended on the hierarchical information model. The basic components of are the elements and the attributes. The elements are describing data, while the attributes are the properties of the elements. The semantic structure of the XML documents is defined through XML schemas from which the individual attributes and elements become perceived in order to be assigned by valid types. The XML Identity Source is depending on a single XML document that is referred to a particular schema of JOSSO. This XML is stored in a file system accessible by the JOSSO server. The entitlements and user records are stored in that XML, while the
user’s credentials are stored in a different file because they constitute sensitive data and must be protected.

18.1 Configuration and installation of an Identity Source

In JOSSO the Identity Sources constitute the basic foundation where the Identity and Service Providers are absorbing data that are used to support authentication processes such as SSO. The properties that characterize an Identity Source are the storage mechanism, the user schema and the access protocol. The first defines the technology involves and the information model (hierarchical or relational) that will be followed. The second defines the structure of the entries. In other words, where the user attributes are referenced and how the semantics of these will be placed. The third determines the messages that the users are going to operate and also the ways of transmitting them over the network.

18.2 Configuration and installation of an Identity Vault

The Identity Vault is an Identity Source that is built in within JOSSO. This Identity Source is based on an Apache Derby relational database and it has the capability to be linked to an Identity Provider as well in a Service Provider. When connected to an Identity Provider it supports authentication processes and when connected to a Service Provider supports identity mapping and account linkage. In order to use this Identity Source, the “Identity Vault” entity is selected from the “Identity Sources” drawer and dragged in the Diagram Canvas. That action opens the following window.

![Create Identity Vault](Picture.37 Create Identity Vault)

In the “Name” field, which is obligatory, the name of the Identity source is written. In the “Description” field, a short description for the Identity Vault is written.
18.3 Configuration and installation of an LDAP Directory Identity Source

The Lightweight Directory Access Protocol (LDAP) is an application protocol for accessing and maintaining distributed directory information services over an Internet Protocol (IP) network. The LDAP Identity Source is accessed through the LDAP protocol which distributes the user entries hierarchal. This type of Identity Source can be linked in both Identity and Service Providers. When connected to an Identity Provider the LDAP protocol is used for authentication processes. When connected to a Service Provider the LDAP protocol is used for local authentication of for increase of the claims towards the Identity Provider. In order to use this Identity Source, the “LDAP Identity Source” entity is selected from the “Identity Sources” drawer and dragged in the Diagram Canvas. That action opens the following window named “Create LDAP Identity Source” which contains the “Base” tab and the “Lookup” tab. The “Base” tab is shown below.

![Create LDAP Identity Source Base tab](image)

**Field Description:**

**Name:** The name that is chosen for the Identity Source.

**Description:** A short description for the Identity Source is written.

**Initial Context Factory:** The default implementation of the Oracle LDAP provider (com.sun.jndi.ldap.LdapCtxFactory).

**Provider URL:** The URL for the LDAP Directory Server. The default URL is “ldap://localhost:389”.

**Security Principal:**

**Security Credential:**
Security Principal: Here is written the security principal for the users’ authentication to the service. The default value for the OpenLDAP\(^{11}\) is “uid=admin, ou=system”.

Security Credential: Here is entered the password for the security principal.

Security Authentication: Here it is defined which authentication mechanism will be followed. There are three options. The first one is “None” for anonymous binding, the second is “Simple” for password authentication and the third is “Strong” for X.509 authentication certificates.

Search Scope: Here is entered the search pattern in order to obtain user and role entries in the LDAP directory. There are four options. The first is “Base”, where the search queries are limited to the specified contexts. The second is “One”, where the LDAP queries search only the immediate children of the LDAP object corresponding to the DN (Distinguished Name) for users and roles. The third is “Subtree”, where the LDAP queries will search the whole directory subtree under the baseDN for roles and users. The fourth is “Children”, where the LDAP queries search directly below to the base entry children.

The “Lookup” tab is shown below.

![Create LDAP Identity Source Lookup tab](image)

\(^{11}\) OpenLDAP Software is a free, open source implementation of the Lightweight Directory Access Protocol (LDAP) developed by the OpenLDAP Project. It is released under its own BSD-style license called the OpenLDAP Public License
Field Description:

User DN: Here is written the Distinguished Name of the user. The default parameters are “ou=People, dc=my-domain, dc=com”.

Principal UID attribute ID: Here is entered the attribute name that contains the users’ identifier. The default value is “uid”.

Role Matching Mode: Here is selected the mechanism for obtaining roles for users. There are three options. The first is “Distinguished Name”, where the roles are retrieved with the usage of the DN as a key. The second is “User ID”, where roles are retrieved with the usage of the user identity. The third is “User Principal”, where roles are retrieved with the usage of the User Principal Name.

UID Attribute ID: Here is entered the attribute identifier that contains the UID. The default value is “uniquemember”.

Role Attribute ID: Here is entered the attribute identifier that refers to the name of the role. The default value is “cn”.

Updatable Credential Attr: In this field is entered the attribute identifier that contains the token value for the Remember-Me authentication method.

Credential Query: Here is entered the query for retrieving the username/password values. The default values are “id=username, userPassword=password”.

User Properties Query: here is entered the query for retrieving the user attributes. The default values are “mail=mail, cn=description”.

18.4 Configuration and installation of a RDBMS Identity Source

A relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model. The RDBMS Identity Source is accessible through the Java Database Connectivity (JDBC) driver. This Identity Source can be linked with both the Identity and Service Providers. The connection with an Identity Provider allows queries to retrieve user records for authentication purposes using as a foundation the JDBC driver database. The connection with a Service Provider allows queries to authenticate the users in a local level or to increase claims towards the Identity Provider. In order to use this Identity Source, the “DB Identity Source” entity is selected from the “Identity Sources” drawer and dragged in the Diagram Canvas. That action opens the following window named “Create DB Identity Source” which contains the “Connection” tab, the “User Lookup” tab and the “Password Update” tab. In the “Name” field is entered the preferable name of the Identity Source. The “Connection” tab is shown below.
Field Description:

Driver: Here is chosen the JDBC-ODBC driver that connects to the database. In order to use this driver for Linux based systems, the corresponding jar file is copied into the $JOSSO2_HOME/lib/jdbc folder.

Connection URL: It is the string sequence that serves the purpose of establishing a linkage to the preferred database.

Username: Here is entered the preferred username that will be promoted to the database during its establishment.

Password: Here is entered the preferred password that will be promoted to the database during its establishment.

Skip Connection Test: This checkbox, if selected, verifies the connection to a database.

Test Connection: This button, if clicked, verifies that JOSSO can reach the database with the usage of the appropriate supplied attributes.

Next, there is the “User Lookup” tab, as it is shown below, where the user can provide the appropriate SQL queries used by the Identity Provider for the retrieval of the roles and credentials of the users.
Field Description:

User Query: It is an SQL query that selects the record from the table of users. It is automatically filled according to the default schema.

Roles Query: It is an SQL query that selects the roles of the records for the selected user. It is automatically filled according to the default schema.

Credentials Query: It is an SQL query that selects the credential records for the selected user. It is automatically filled according to the default schema.

Properties Query: It is an SQL query that select customized attributes of the user that subsequently will be conveyed as claims in authentication assertions. It is automatically filled according to the default schema.

Test Queries: This button, if clicked, ensures that the SQL queries are that they are written correctly and return the awaited information.

In the third tab, named “Password Update” and is shown below, the user can provide SQL queries that are for self-services.
Field Description:

Credentials Update: It is an SQL update query that allows the change of the users’ credentials. It is automatically filled according to the default schema.

Relay Credentials: It is an SQL update query that updates the users’ credentials based on the Remember Me functionality. It is automatically filled according to the default schema.

Test Update: This button, if clicked, ensures that the SQL update queries are written correctly and return the awaited information.

18.5 Configuration and installation of a XML Identity Source

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It is defined in the XML 1.0 Specification produced by the W3C, and several other related specifications, all gratis open standards. The XML identity source is one that is based on a hierarchical information model and its basic components are elements and attributes, where elements describe data and attributes are the properties of elements. The XML identity source needs to be based on a XML document that corresponds to a schema specified for JOSSO, and must be in a folder accessible by the JOSSO server. Such a XML document is shown below.
<josso-users>
  <users>
    <user>
      <name>user1</name>
      <properties>
        <property>
          <name>user.name</name>
          <value>User 1 Name</value>
        </property>
        <property>
          <name>user.lastName</name>
          <value>User 1 Last Name</value>
        </property>
        <property>
          <name>user.registrationDate</name>
          <value>2004/09/11</value>
        </property>
      </properties>
      <roles>
        role1, role2
      </roles>
    </user>
  </users>
</josso-users>
<role>
  <name>role2</name>
</role>
</roles>
</josso-users>

And next is an XML document that contains the users’ credentials.

<josso-credentials>
  <credential-set>
    <key>
      user1
    </key>
    <credential>
      <name>username</name>
      <value>user1</value>
    </credential>
    <credential>
      <name>password</name>
      <value>7ea2bd72bfc7dabfdce0b5760ebcf52</value>
    </credential>
    <credential>
      <name>userCertificate</name>
      <value>-----BEGIN CERTIFICATE-----
MIIDjjCCAvegAwIBAgIBAjANBgkqhkiG9w0BAQQFADCByjELMAkGA1UEBxMDTi9BMRswGQYDVQQK
ExJKT1NTTbyBPcmdh
mzby5vcmcx
-----END CERTIFICATE-----</value>
    </credential>
  </credential-set>
</josso-credentials>
In order to use this Identity Source, the “XML Identity Source” entity is selected from the “Identity Sources” drawer and dragged and dropped in the Diagram Canvas. That action opens the following window, named “Create XML Identity Source”.

-----END CERTIFICATE-----
Field Description:

Name: Here is entered the name of the XML identity source

Description: A short description for the XML identity source is written.

XML URL: Here is entered the folder location of the XML document that possesses the details and entitlement entries of the user.

19. Connections

A Connection is the chain that puts together all Identity Appliance elements that are needed in order to create and execute the Identity Appliance. Connections are used to join two Identity Appliance elements. Each connection enables a flow of data and control from one Identity Appliance element to another, creating a contract between the two elements.

The connection types that JOSSO uses are described subsequently.

19.1 Federated connection

A Federated connection is used to connect an Identity Provider with a Service Provider and create a Circle of Trust. This circle of trust provides users with enhanced operational efficiency.

19.2 Activation

Activation is a connection used to connect the Service Provider to an execution environment, such as Apache Tomcat or JBoss, applying SSO semantics to execution environments hosting Service Providers for a specific Identity Appliance. The execution environment element defines the SSO agent that will be provisioned for bringing forward the SSO capabilities for the associated Service Provider.
19.3 Identity Lookup

The Identity Lookup connection defines, as an entity, whether the Identity Source is an Identity Provider or a Service Provider. It supports identity and access management processes such as authentication and authorization processes. An Identity Lookup that connects an Identity Provider with an Identity Source defines which Identity Source meant for authorization will be used, and what ways will be implemented in order to access the source to obtain the user credentials that are necessary for the authentication. Furthermore, it specifies the authorization source from which user and entitlement details will be obtained. An identity lookup that connects a Service Provider with an Identity Source determines which local Identity Source, that is used for authorization purposes, is going to be executed and with what means it will access it.

19.4 Identity Verification

The Identity Verification connection is used to pass the authentication process of an Identity Provider to an authorized third party entity. Such an entity could be an Authentication Server that can be associated with an Identity Provider.

20. Privilege and Account Management

In this section we are going to describe how we can create users and groups that are stored for usage from the Identity Vault element. Both accounts and groups can be provisioned. Accounts can also be associated to one or more groups in order to serve as the input for Role-Based Access Control (RBAC). We can create groups and users after we have firstly created the Identity Appliance. Users and groups are being created through the Account and Entitlement Management screen. This screen gives us the opportunity to work with users and groups.
If we want to create a user or a group we have to follow the same steps with a small difference, when we want to create a user we choose from the options that we have available from the Manage Users button, as illustrated in Picture: 45, additionally, when we want to create or edit a group we choose the Manage Group button as it is illustrated in Picture: 46.
20.1 User Creation

Step 1
Click on Manage Users button. Clicking Manage User button brings all the options that we have for working out with users. The options that we have, as they are illustrated in the following picture, are create, edit, delete or search a user.

Step 2
Click on Create User button.
After clicking on this button it appears the window that is illustrated in the following picture:

![Create User General tab](image)

**Picture 49 Create User General tab**

As we can see in Picture: 49, the “Create User window” consists of the following tabs General, Preferences, Groups, Security and Password. In the first tab, “General”, which is the first tab we see we have to fill in elements that will characterize our user. The elements Username, First Name and Last Name which have a red asterisk are mandatory, the other elements as for example E-mail Address are optional.

In the second tab “Preferences”, as illustrated in Picture: 50, we can choose the language.
In the third tab “Group” if we want the user that we have created to be a member of a group we can choose the group we want for him from “Available Groups” list and drop it in the Member of list, as illustrated in Picture : 51.
The forth tab “Security” is illustrated in Picture: 52

![Create User](image)

Picture.52 Create Group tab

In this window we can choose options that are related with the security of the user, for example we can choose the expire account checkbox in order to define the expiration date of his credentials, or after a limited simultaneous login tries, the user will not have access in his account.

The last tab “Password” is illustrated in Picture: 53
In this tab we have the opportunity to give to the user a password with which he will have access to his account. Furthermore through this tab we can determine the rights that the user will have according with his password. More specifically we can give to the user the opportunity to change his password by himself and the most important from the security point of view is that we can define how often the user should change his password. Moreover we can choose to generate automatically a password if we will not choose one by ourselves and finally we can choose if we want to disclose the user from getting his new password through email.

After we fill in all the elements that we want we click on save button and we have easily created our user.
The user that was just created will appear in the Account & Entitlement Management console as illustrated in the following picture:

*Picture.55 Account & Entitlement Management 1 Screen*

At the bottom of this window is the “Identification” menu. If we select a user that we have already created in this field we can see all the elements that we have chosen for this particular user by clicking in the right the tab we want to see.

Furthermore in this screen, as already we have mentioned, we can delete or modify a user. If we want to delete a user we have to click on this particular user so as to be selected and then we have to click on Delete User button.

*Picture.56 Delete User*
Then will appear the window that is illustrated in Picture 57 which applying for confirmation of deletion of this particular user.

![Picture 57 Confirmation of Deletion Window](image)

In the same way if we want to modify a user we click on the particular user so as to be selected and then we click on Edit User button.

![Picture 58 Edit User Button](image)

After clicking this button it is appearing the window that illustrated in Picture: 59. In this window we can make any changes we want in this particular user.
The last button, that will be mentioned, is the Search button. This button gives us the opportunity to find easily a user that we have created. This option will be useful if we have created many users. If we click on Search button is appearing the window that is illustrated in Picture: 60.

**Picture.59 Edit User Window**

In this window we have to fill in the field that characterizes the user that we want to search, for example his Username, by clicking on the Search button. In this way will appear only this user in the Account & Entitlement Management screen. After we find
the user that we want, we can click on the Remove search filters button and all the users that we have created will reappear.

**Picture.61 Account & Entitlement Management 2 Screen**

**Picture.62 Remove Search Filters Button**
20.2 How to create a Group

As we have already mentioned, through the Account & Entitlement Management screen, apart from creating Users we can also create in the same way Groups in which a user can be a member. The main difference is that we have to click on the Manage Group button, when we click on this button it brings all the options that we have for working with groups. The options that we have, as they are illustrated in Picture: 46 are creating, modify, delete or search for a group. As we want to create a new group we click on Create Group button and appear the following window.

![Create Group Window](image)

In this window we have to fill in a name for our Group. The “Name” element is mandatory but if we want we can write additionally a description for the Group that is going to be created, this element is optional. After that we click on save button and our Group is created and is appears in Account & Entitlement Management screen as is illustrated in Picture: 64.
Furthermore as we already have mentioned, we have also the ability to modify, delete or search for a Group that we have already created.

If we select a Group and click on Edit Group button is appears the window illustrated in Picture: 65. In this window we can make changes in this particular Group.

**Picture.64 Create Group1**

**Picture.65 Edit Group Window**
In the same way, if we want to delete a Group we have to select the specific Group and click on Delete Group button.

Then it will appear a window, as is illustrated in the following picture, in which we have to confirm the deletion of our choice.

In conclusion if we want to search for a Group we click on Search button. This button gives us the ability to find easily a Group that we have created. This will be useful if we have created many Groups. If we click on Search button it appears the window that is illustrated in Picture: 68.

In this window we have to fill in a field that characterizes the Group that we want to search for, as for example “Name”, and click on Search button, in this way will appear only the specific Group in Account & Entitlement Management screen.
After we find the Group that we want we can click on Remove search filters button and all the Groups that we have created will appear again.

20.3 How to add an extra attribute

The last button that we can use in the Account & Entitlement Management screen is the Manage Schemas button. This button gives us the ability to add an extra attribute that we want for a User or a Group to have and is not included by default. In the following example it will be described how we can add the extra attribute “Email” in a Group. In the same way we can add an extra attribute in a user.
Firstly we choose between User and Group options, after that it appears the proportionate options which are create, modify or delete an attribute. If we click on “New” attribute button it appears the window that is illustrated in Picture: 72 in this window we have to:

- Give a “Name”
- Select the type of the new attribute as for example: string
- Select if we want this attribute to be mandatory and
- Select if we want this attribute to be multivalued
Then we click on save button and the new attribute is appearing as a new tab in the Group entity and as a new field in User entity as illustrated in Picture: 73.

*Picture.72 Add Schemas Attribute*

*Picture.73 Extra Attribute*
21. Identity Appliance Lifecycle Management Screen

When the Identity Appliance is created the next step is to bring this Identity Appliance to life. We can achieve that through the Identity Appliance Lifecycle Management screen. This screen as it is shown in the following picture is consisting of four columns: Saved, Staged, Deployed, and Disposed.

![Identity Appliance Lifecycle Management Screen](image)

Each column represents the different state that the Identity Appliance artifact can be in. To activate our Identity Appliance we have to drag and drop the Identity Appliance gradually from each column to the other so that finally to place it in the Deployed column.

The first column is Saved, and inside it are available all the Identity Appliances that we have created. Furthermore in this state we have the ability to modify or remove the Identity Appliance using the following buttons , respectively.

If we want to continue we have to drag and drop the Identity Appliance in the second column which is called Staged.
If the building operation is successful, the target Identity Appliance will appear in the Staged column as we can see in the following picture.

The Staged column keeps entries for the Identity Appliances that have been successfully compiled. In Staged column we have the ability to edit, rebuild and delete the Identity Appliance. If we click on edit button we automatically transferred in the Identity Appliance Modeler screen in which we can make changes in this Identity Appliance. If the changes are made we can save them and then click on rebuild button . Finally if we click on delete button the Identity Appliance will automatically transferred into the Disposed column.
In order to deploy an Identity Appliance, we select and drag the row for the target Identity Appliance item from the Staged column and drop it onto the Deployed column. Then the Identity Appliance appears in the Deployed column of the Lifecycle Management Window.

As the Identity Appliance is in Deployed column we can click on Start button so the Identity Appliance will be executed or click on stop button to stop the execution for Identity Appliance as we can see below.

An important issue which it should be referred to is that we can have more than one Identity Appliances to executing at the same time.
Another action that we can do in Deployed column is to delete the Identity Appliance by clicking on delete button . If we delete the Identity Appliance it is transferred automatically in the Staged column, as it is shown below.

**Picture.79 Return to the Staged Column**

If we delete it also from the Staged column then the Identity Appliance is transferred automatically in the Disposed column. In this column exist the Identity Appliances that we have deleted and the only action button that we have is delete, if we click on this button the Identity Appliance will be permanently deleted, as we can see below.
Single Sign-On

Picture.80 Disposed Column

Picture.81 Delete the Identity Appliance

Picture.82 Identity Appliance Lifecycle Management 1
22. System Setting

In system setting screen we can see only the user’s license if we are using JOSSO C.E., but if we use JOSSO E.E., through this screen we can download new updates. Furthermore we can create our notification scheme, for example we can define in which email we will receive warnings. Finally through this screen we can configure the SMTP settings.
In the following example we are going to describe how we can create an Identity Appliance, which will provide SSO services. To achieve SSO we should have an Identity Appliance which will consist of an Identity Provider which will be connected with at least two Service Providers. Each service Provider should trust the Identity Provider, which is why we have to connect them with a Federation connection. Furthermore in the following example we will use an Identity Vault which will be connected with Identity Appliance with an Identity Lookup connection. Finally each Service Provider will be executed in an execute environment. In this example we are going to use Apache Tomcat and JBoss but JOSSO gives us others environment solutions as well. It is important to mention that in order to run correctly the Identity Appliance that we going to create, we should download Apache Tomcat and JBoss web containers, which should run in parallel with the Identity Appliance.

**Step 1**

Create the Identity Appliance

Choose an Empty Identity Appliance on the action bar and then click on “New” button.
At this point we have to mention that we have another choice in the drop list which is “Federated SSO baseline”. If we choose this option it opens a window that we can give names in the entities that we want to use and the Identity Appliance is ready. We do not have to drag and drop the items from Pallet to Diagram Pallet and connect them all with each other. But in the following example we choose to use the “Empty Identity Appliance” option as we have already mentioned.

After we click on “New” button it appears the “New Empty Identity Appliance” window. The “New Empty Identity Appliance” window is consisting of the following fields:

- **Name**: a unique identifier for the identity appliance
- **Description**: a description for the identity appliance
- **Realm Name**: provides a unique name for the elements that exist in the Identity Appliance
- **Appliance Location**: it defines the host name and port where identity endpoints for the Identity Appliance are going to be bound
As we have filled in all the mandatory fields we click on Next button and appear the “New Empty Identity Appliance” window that illustrated in the following picture:

![Confirm Identity Appliance Window](Picture.88)

_in this window we can see the options that we have made in the previous step. In this window we have three options. We can click on, “Confirm” button that creates the Identity Appliance or click on “Previous” button and return to the previous window to make changes or click on “Cancel” button which canceled each action that we do. So we click “Confirm” button and the Identity Appliance is created._

![Identity Appliance Modeler Window](Picture.89)
Step 2

Create the Identity Provider.

In order to create the Identity Provider we drag from Pallet the Identity Provider entity and drop it into Diagram Canvas.

*Picture 90 Entities*

As we drop the Identity Provider into the Diagram Canvas it appears the “New Identity Provider Definition” window, where we can choose the name of the Identity Provider and to give a short description.

*Picture 91 Identity Provider Care tab*
As we can see from picture: “New Identity Provider Definition” window consists of three tabs:

- Core
- Contract
- Certificate

In “Core” tab we can specify the reachable Identity Provider endpoints for the users. A default location is created based on attributes given at the creation time of the appliance. It is recommended that this location must stay as-is in order to avoid inconsistencies later on. In “Core” tab we have to fill in the following fields:

**Location**: Here there are determined the access protocol (http or https), the host name, the port and the path that bonds the endpoints of the Identity Provider.

**Authentication Contract**: This is the mean for submitting claims to the Identity Provider. It is based on specific predefined JOSSO parameters, that are submitted when it is attempted a web-based authentication method.

**Authentication Mechanism**: It describes the authentication process of the users. It has only one option, the “Simple Authentication” mechanism which is based on the username – password method.

**Authentication Assertion Emission Policy**: This option defines the way assertions are issued during a successful authentication process. These assertions are transferred through security tokens to the relying parties.

The next tab is the “Contact” tab. Here there are determined the SAML Profiles and Bindings, but also the security level of the artifacts that are exchanged between the Identity Provider and the Service Provider. The “Contact” tab is shown below.
The following fields we see on “Contract” tab:

Enabled SAML Profiles: The SSO checkbox represents the “Web Browser Single Sign-On Profile”. This is the most important SAML profile. The SLO checkbox enables the Single Logout Support option.

Enabled SAML Bindings: Here it is determined the schema of the SAML protocol based on standard formats and other communication protocols. The Http Post checkbox relays the SAML messages through a HTTP POST method. The Http Redirect checkbox relays SAML messages through a HTTP GET method. The Artifact checkbox relays SAML messages through the SAML Artifact Binding, which is based on HTTP Redirect and SOAP bindings. Its purpose is to facilitate the exchange of SAML messages. The SOAP checkbox relays SAML messages through the SOAP protocol over HTTP/HTTPS.

Sign Authentication Assertions: In this field it is decided whether the SAML Authentication Assertions are digitally signed or not. The digital signing provides guarantees of integrity and identity of the Service Provider towards the Identity provider.
Encrypt Authentication Assertions: If this checkbox is chosen it means that the SAML Authentication Assertions are encrypted. This feature provides confidentiality to the exchangeable SAML messages, prohibiting eavesdropping attempts.

Third and last tab is the “Certificate” tab. Here the user can select between the default JOSSO keystore and an uploaded one, in order to keep the private and public key pairs that are used to secure the exchange of the SAML messages among the Identity Provider and the Service Provider. If the Uploaded keystore file option is chosen, the user must set up the components of the trust system himself, which is based on the public key infrastructure (PKI). This system allows provisioning of confidentiality, integrity, authentication and non-repudiation. This tab is demonstrated below.

![Picture 93 Identity Provider Certificate tab](image)

The following fields we see on “Certificate” tab:

Use Default Keystore: This option is recommended only for sandbox settings where security is not an issue.

Use the keystore file: Usage of a customized keystore file.

Certificate Key Pair: This option gives the ability of uploading a keystore file from the user’s local file system.
Format: This indicates the format of the keystore file. In JOSSO the only supported format is Java Keystore (JKS).

Keystore Password: In this field a password is typed that the Identity Provider will use in order to decrypt the keystore and gain access to the key pair entities.

Certificate Alias: It is the identifier of the keystore entry for the Identity Provider’s public key.

Key Alias: It is the identifier of the keystore entry for the Identity Provider’s public key.

Key Password: It is the password field for obtaining the keystore entry which keeps the Identity Provider’s private key.

If the above steps and the corresponding fields are selected accordingly, the user clicks on the OK button to create the Identity Provider entity.

![Identity Appliance Modeler with Identity Provider](image.png)

*Picture.94 Identity Appliance Modeler with Identity Provider*

As we have created the Identity Appliance we can delete it by clicking on the delete button.
In this example we do not make any changes in the default.

**Step 3**

Create Identity Vault.

In order to create Identity Vault we drop “Identity Vault” item from Identity Sources from Pallet and we drop it into the Diagram Canvas. Then appears the “Create Identity Vault” window in which we have to fill in the Name that will be the identifier for the Identity Vault and if we desire we can give a description.

![Identity Sources](Picture.95 Identity Vault)

Create Identity Vault

<table>
<thead>
<tr>
<th>Name: users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
</tr>
</tbody>
</table>

![Identity Vault](Picture.96 Create Identity Vault)

As we have chosen Identity Vault we will use the Identity Lookup connection from the “Connections” drawer, in order to connect the Identity Vault with the Identity Provider.

![Identity Lookup Connection](Picture.97 Identity Lookup Connection)
Step 4

Create the Service Provider.

From the “Entities” drawer we click on the “Service Provider” entity. Then we drag it to the preferred place in the Diagram Canvas. This action opens a window named “New Service Provider Definition”, as it is shown below, where the name of the Service Provider can be putted.

![Service Provider](image)

Picture.98 Service Provider

The “New Service Provider Definition” window is consisting of three tabs:

- Core
- Contract
- Certificate
At the “Core” tab is specified the location of the Service Providers’ endpoints that the users will use. This location is created using attributes provided during the creation of the Identity Appliance. It is recommended not to change them for consistency reasons.

Core tab has the following fields:

**Location:** Here the access protocol is defined. It is either http or https. Moreover, there are defined the host name and the context path of the Service Providers’ endpoints. The users can access the desired services of the Service Provider using URIs capable of using this location value. It is advised the usage of a valid host name in order to allow the Identity Appliance services to be disconnected from a physical host.

**Account Linkage Policy:** This option gives the means that will be used from the input claims in order to be transmitted in the security token and to be mapped to output claims, which then are issued and submitted by the Identity Provider. After that step, the claims are outlaid by the relevant party in order to grand access to the authorized users. The Account Linkage Policy has three alternative options. The first is the “Use Theirs” option where it is used to connect Identity Provider and Service Provider accounts utilizing the given name identifier and mapping input to output claims one to one. The second is the “Use Ours”, where is again used to connect Identity Provider and Service Provider accounts utilizing the given name identifier but the output issued claims are based only on the user details that are disposable in the Identity Source linked to the Service Provider. The third is the “Aggregate” option, where the Identity Provider and Service Provider are linked as before but the output claims are merged in the user details transmitted in the security token and to those that are acquired from the linked Identity Source to the Service Provider.

The next tab is the “Contract” tab. Here the SAML profiles and SAML bindings are enabled used to interact with the Identity Provider. It is also possible to check the level of security the artifacts use that are responsible for the exchanged messages between the Identity and Service provider, as it is shown below.
Contract tab has the following fields:

**Enabled SAML Profiles:** The SAML profile that is most significant, is the “Web Browser Single Sign On Profile”, which is enabled by selecting the SSO checkbox. If the Single Support Logout option is required, it is enabled by checking the SLO checkbox.

**Enabled SAML Bindings:** This option specifies the mapping of the SAML protocol message, towards standardized message formats and other communication protocols. If the Http Post checkbox is selected the SAML messages are transmitted through the HTTP POST method. If the Http Redirect checkbox is selected the SAML messages are transmitted through the HTTP GET method. If the Artifact is selected the SAML messages are transmitted through the SAML Artifact Binding, which is based on HTTP Redirect and SOAP bindings in order to exchange SAML messages. The last checkbox is the SOAP checkbox. If selected the SAML messages are transmitted through the SOAP protocol over HTTP/HTTPS.

The third screen is the “Certificate” tab. Here the keystores of the private – public key pairs are selected that secure the exchange of the SAML messages among the Identity and Service Provider. A trust system is created based on the public key infrastructure (PKI), providing confidentiality, integrity, authentication and non-repudiation among the trusted entities. The “Certificate” tab is shown below.
Certificate tab has the following fields:

**Use Default Keystore:** This option is recommended only for sandbox settings where security is not important. In computer security, a sandbox is a security mechanism for separating running programs. It is often used to execute untested code, or untrusted programs from unverified third-parties, suppliers, untrusted users and untrusted websites.

**Upload the keystore file:** This option exists if the user has a customized key pair and wants to use it instead the default keystore.

**Certificate/Key Pair:** This option grants the user to select a desired keystore file from his local file system.

**Format:** In this option the only supported keystore format in JOSSO is the Java Keystore (JKS).

**Keystore Password:** The required password that is needed to open the keystore files in order to obtain the necessary key pair for secure SSO message exchanges.

**Certificate Alias:** It plays the role of an identifier of the keystore entry for the public key.
Key Alias: Here is the name of the keystore entry that is used to acquire the corresponding private key.

Key Password: It is the necessary password for acquiring the private key.

As we have created the Service Providers we have to connect each one separately with Identity Provider through a Federation Connection which will create a trust relationship between the Identity Provider and the Service Providers.

The Federation Connection can be found in the “Connection” drawer from the Palette.

As we drag the item from Pallet and drop it into Diagram Pallet it appears the “Create Federated Connection” window. This window consists of two tabs:

- Identity Provider Channel
- Service Provider Channel

By default when it appears the “Create Federated Connection” window it has already a Name although we can add a description if we desire.

Identity Provider Channel has the choices that we have already made when we create the Identity Provider and Service Provider Channel has the choices that we have already made when we created the Service Provider.

As we have created the Service Provider and established a circle of trust with the Identity Provider the last thing that we have to do is to choose an execution environment where Service Providers will run in order to offer services to end users and applications. In this example we are going to use the Tomcat 6.0.35 and JBoss 5.0.0.

Tomcat and JBoss web containers can be found in “Execution Environment” from Pallet. As we drop it to Diagram Canvas it appears the “Create Tomcat Execution Environment” window and “Create JBoss Execution Environment” that illustrated in the following pictures.
In the above windows we have to fill in the mandatory fields and to select the check boxes (for more information look on chapter 15.11 and 15.12 respectively).
The last thing that we have to do is to activate Tomcat and JBoss, so in order to perform the activation we click on “Activation” item from “Connections” in Pallet and then we click on Tomcat and the Service Provider in which we want to connect, and it appears the “Create JOSSO Activation” window, illustrated in the following picture:

In these windows the mandatory fields which are Name, Partner Application Identifier and Partner Application Location are supplemented. The only change that may be needed is in the port and to add partnerapp which is the location that SSO will be take place.

As it is illustrated in Picture: 106 the same window appears when we are going to connect JBoss with a Service Provider.
Picture 106 Activate JBoss

In this example Tomcat uses port 8082 and JBoss 8080.

It is important to mention that after completing the activation we have to reactive Tomcat and JBoss. To make this action we have to click on Tomcat and JBoss separately each time and click on the Activate button, after we have checked the Reactivation check box. With this action we “Reactivate” the execution environments and all the previous set up is been deleted.

Picture 107 Reactivate window

Step 5
Create User and Group.

We create a user with the username: user1 which will be part of role1 group as illustrated in the following picture:
As the Identity Appliance was designed in the “Identity Appliance Modeler” the next step is to bring the Identity Appliance into life through the “Identity Appliance Lifecycle Management” window. In this window we have to drag and drop our Identity Appliance from Saved Column into Staged Column and finally in Deployed column and click on start button so the Identity Appliance run.

Picture.108 Users and Groups

Picture.109 Bring Identity Appliance into Life Saved Column
While the Identity Appliance is running we open a browser and we type the location from Service Provider which is: [http://localhost:8082/partnerapp](http://localhost:8082/partnerapp) as it is illustrated in the following picture:

![Picture.111 Atricore Login window](image)

*Picture.110 Bring Identity Appliance into Life Deployed Column*

*Picture.111 Atricore Login window*
As the page is loaded it appears the site of Atricore, which is stored locally, and we click on “Login” button. After that it appears the field Username and Password. In these fields we type in the username and password of the user that we have already created in Atricore and click on login button.

Then it is loaded the login page.

After that step, we have logged in and the following picture appears.
If we click on “protected” option the following picture appears.
As we have just logged in the first Service Provider, we open another browser and we type the location of the second Service Provider which is: [http://localhost:8080/partnerapp](http://localhost:8080/partnerapp) and it is loaded the login page without the need to authenticate the user again.
Single Sign-On

*Picture.116* [http://localhost:8082/partnerapp](http://localhost:8082/partnerapp)

*Picture.117* Loading Page 1

*Picture.118* Loading Page 2
**Picture 119 Login with SSO**

This is a very simple JOSSO partner application, you’re accessing a protected web resource. You can find some tips to use in your own app!

**Picture 120 User Information**

This is a very simple JOSSO partner application, you’re accessing a protected web resource. You can find some tips to use in your own app!
24. Cloud computing in JOSSO

JOSSO2 Enterprise Edition (Josso 2 E.E.) supports Cloud with the following applications Google apps, Salesforce and SugarCRM. In the following example it will be used Google apps and Salesforce to demonstrate how the Cloud technology works along with JOSSO2 E.E. To bring the example into life the Domain name: www.cloudjosso.com is created and an account for each application. In both applications it is used the same username which is josso@cloudjosso.com but with different passwords. As long as the prerequisites are defined, the Identity Appliance in JOSSO2 E.E. is created, which consists of an Identity Provider, Google apps and Salesforce from the Cloud Entities and an Identity Vault from Identity Source category as is illustrated in the following picture.

![Picture.121 Cloud](image)

Both Cloud Entities and Identity Vault should be connected with the Identity Provider, each Cloud entity will be connected with a Federation connection and Identity Vault will be connected with an Identity Lookup connection. In the following steps it is described the above Identity Appliance.
We should mention that in the following example we will not make any changes into default values.

**Step 1**

Create an Identity Appliance.

![Image of Identity Appliance](Picture.122 Create Identity Appliance for Cloud Example)

![Image of Identity Appliance](Picture.123 Conform Identity Appliance for Cloud Example)
Step 2

Create the Identity Provider.

Picture 124 Create Identity Provider for Cloud
Picture.125 Identity Provider Contact tab for Cloud
Step 3

Create the Identity Vault.

![Create Identity Vault for Cloud](image1)

*Picture 126 Create Identity Vault for Cloud*

Step 4

Create the Salesforce Entity.

![Create Salesforce Entity](image2)

*Picture 127 Create Salesforce Entity*
Step 5

Create Google app Entity.

*Picture.128 Create Google app Entity*

Step 6

Create the user josso@cloudjosso.com which is belongs in the role 1 Group.

*Picture.129 Create the user josso@cloudjosso.com*
Step 7

Start executing the Identity Appliance.

![Image of Identity Appliance](image1.png)

_Picture 1.30 Start Running the Identity Appliance_

As the Identity Appliance is ready and it is running the next step is to configure Google Apps and Salesforce to work as Single-Sign On environment. Before logging in Google apps and Salesforce in order to make the configurations, we have to save the Identity Provider certification which is in pem format and the metadata file which is in xml format and contains all the links that will be needed to complete the configuration for the single sign on.

![Image of Exporting Metadata](image2.png)

_Picture 1.31 Certification_
Both files are in Josso Identity Provider. To begin with, Identity Provider certification it is located in certification tab and metadata is located in contract tab. As we have those files we have all the information we need to make the configuration.

Firstly it will be described the configuration of Google apps with 6 steps:

Step 1
Login in Google app application (http://www.google.com/enterprise/apps/business)
Picture 133 Login in Google app Application

**Step 2**
Click on Advanced tools menu

Picture 134 Advanced Tools Menu

**Step 3**
Click on Set up Single Sign-On (SSO) option which is on Authentication category
Step 4

Click on Enable Single Sign-On option and then complete the Sing-in page URL, Sing-out page URL and check the option use a domain specific issuer.

The URL that it is needed is on the metadata file as it is illustrated in the following pictures.

### Picture 135 Set up Single Sign-On

![Set up Single Sign-On](image)

**Step 4**

Click on Enable Single Sign-On option and then complete the Sing-in page URL, Sing-out page URL and check the option use a domain specific issuer.

The URL that is needed is on the metadata file as it is illustrated in the following pictures.

### Picture 136 Metadata

![Metadata](image)
Picture 137 Sign-out Page URL

Picture 138 Set up Sign-out
Step 5

Click on save change button

![Save changes Button](Picture.139 Save Changes Button)

Step 6

Logout of the application

In the same way is configured the Salesforce application.

Step 1

Login in Salesforce application ([https://login.salesforce.com/?locale=eu](https://login.salesforce.com/?locale=eu))
**Picture 140 Login in Salesforce Application**

**Step 2**

Click on set up option

**Picture 141 Setup Salesforce**
Step 3
Click on Security controls options which are on Administration set up category.

Picture.142 Security Controls on Salesforce

Step 4
Click on Single Sign-On Setting option.
Step 5

Click Edit Button.

Picture.143 Single Sign-On Setting Option on Salesforce

Picture.144 Edit Button
Step 6

Check SAML Enabled option.

![Picture 145 SAML Enabled Button]

Step 7

Choose SAML Version 2.0 from the list.

![Picture 146 SAML Version]
Step 8

Click Choose file button and attach the Identity Provider certification file and in issuer option fill in the link from metadata file as it is illustrated in the following picture.

Picture.147 Identity Provider Certification
Step 9
Click “Save” button.

Step 10
Logout from the application.
As the applications have been configured it is time to try the Single Sign-On technology. To do that, a new tab on Firefox is opened and it is typed the following URL:


Then it will load the page to login to JOSSO2 E.E as illustrated in Picture: 150

In this page we login with the username and password of the JOSSO user that it had been created in Identity Vault in JOSSO2 E.E. After login button is clicked, it automatically redirects the user in the Salesforce application.
Single Sign-On

Picture 150 Login Page

Picture 151 JOSSO Processing
After that step, we will try to access Google Apps, so we open a new tab in Firefox and choose to open the email account.

As we click on Go button we automatically have access in Google Apps without filling in the username and password.
Picture 154 Automatically Login

Picture 155 Login on Cloud mail
25. Two-Factor Authentication with WiKID

JOSSO2 Enterprise Edition, by default authenticates a user by username and password, although we can choose to use a two factor authentication using WiKID Authentication System. WiKID Strong Authentication System consists of three parts: the WiKID server, the WiKID token client and a network client. The WiKID server is written in Java.

If we want to use WiKID Authentication System in JOSSO2 Enterprise Edition we have to create an Identity Appliance which consists of an Identity Provider and at least two Service Providers, which will be connected with Identity Provider through Federation connection. Furthermore, each Service Provider will use an execution environment like Apache Tomcat or JBoss. Additionally, we will use an Identity Vault which will be connected to the Identity Provider.

In order to use WiKID Authentication System we have to configure first the WiKID Authentication System and then to connected it with JOSSO2 through the Diagram Canvas.

The configuration of WiKID Authentication System is described with the following steps:
**Step 1**

We download the WiKID Server .deb file from http://www.wikidsystems.com/ and the Software Token Clients.

**Step 2**

We have to install prerequisite programs with the following Linux commands:

- `sudo apt-get install yum`
- `sudo apt-get install`
- `ln -s /usr/lib/jvm/java-6-openjdk/jre/bin/keytouls`
- `export JAVA_HOME=/usr/lib/jvm/java-6-openjdk/`
- `sudo dpkg -i wikid-server-xxx.deb`
- `sudo apt-get install postgresql openjdk-6-jdk openjdk-6-jre openssl libwww-perl sudo locate ntpdate`

After we have installed WiKID Server we type the following command to continue with the setup

- `sudo /opt/WiKID/bin/wikidctl setup`

finally we type

- `sudo /opt/WiKID/bin/wikidctl start`

and the server is executed.

---

12 So as to execute this command we have to be on folder which contains wikidctl file
As the WiKID server is executed we open a browser and login in WiKID. The default username and Password are WiKIDAdmin and 2Factor respectively. After that we login in on WiKID Server and we can continue with the configuration.

**Picture.157 WiKID**

**Picture.158 WiKID Login**
**Step 3**

Create an Intermediate CA clicking on “Create an Intermediate CA” link.

**Picture.159 Configure WiKID**

**Picture.160 Create an Intermediate CA**

In the window that appears we fill in all the fields.
The most important field is the “Passphrase” because this Passphrase will be used when we will connect JOSSO with the WiKID Authentication System. As we have filled in all the fields we click on Generate button and appear a window with the certificate that we just created.

---

**Picture 161 CA fields**

---

**Picture 162 Generate Button**
As a next step we copy this certificate and click on the following link: [https://ca.wikidsystems.com/wikid/newcertreq.jsp](https://ca.wikidsystems.com/wikid/newcertreq.jsp) that opens a pop up window in which we paste the certificate and click on submit for processing button.

Thank you. Your WiKID intermediate certificate is being generated....

Thank you for trying WiKID.

If you choose to use the commercial version after 30 days please contact [certs@wikidsystems.com](mailto:certs@wikidsystems.com) for a permanent production certificate.
Then we copy the certificate from the pop up window and paste it to the empty frame in WiKID in order to install this certificate.

![Install Certificate](image165)

After we paste it, we fill in the passphrase field and click on Install Intermediate Certificate button. In this way we just have installed the certificate.

![Install Intermediate Certificate Button](image166)

**Step 4**

Create a localhost Certificate by clicking on “Create a localhost Certificate” link. In this window we fill in all the fields and click on generate button.
Single Sign-On

Picture.167 Create a localhost Certificate

Picture.168 localhost Certificate Fields
**Step 5**

Enable network protocols.

WAUTH is already enabled but it is recommended to enable and another protocol.

**Step 6**

Create a Network Client.
As we have created a Network Client it has been created a certificate. This certificate will be uploaded in JOSSO when we go to connect JOSSO with WiKID Authentication System.

**Picture.171 Network Client**

**Step 7**

Create a Domain.

**Picture.172 Create a Domain**
Step 8

Create a user.

Before clicking on manually validate a user button we have to open a WiKID token in order to create a registration code which will characterize the user that we going to create.

*Picture.173 WiKID token*

When we will try to login in JOSSO with the username of the user that we have just created, it will be authenticated by the passphrase that WiKID token creates.

*Picture.174 Registration*

When we have configured WiKID Authentication System, we have to connect it with JOSSO. In order to connect JOSSO with WiKID Authentication System we have to upload the two certificates that we have created in WiKID Authentication System to the Atricore Console. The certificate, as we have already stated, is in Network Client
and we have to download it. The other certificate is located in opt/WiKID/private folder and is a CACertStore certificate. As long as we have available these certificates we can continue with the JOSSO setup.

As we already have created the Identity Appliance as we have seen in chapter 23 the next step that we have to do is to add WiKID entity by clicking on WiKID item from pallet and drop it into Diagram canvas.

![Picture 175 Add WiKID in JOSSO](Image)

As we drop the WiKID item into Diagram canvas the window that illustrated in picture 176 appears. In this window we have to fill in the fields and to upload the certificates that we have already mentioned (for more information look on chapter 17.3).
After that step we click on OK button. The last thing that we have to do is to connect the Identity Provider with WiKID with an Identity Verification connection.

The only difference that worth mentioned in this authentication process is that when we are going to login in Atricore we have to give a Username and a Passphrase instead of a Username and a Password, so we have to generate a passphrase with WiKID token and fill with that the passphrase field.
Picture.178 Login with Passcode

Picture.179 Get Passcode
26. Conclusion

In this thesis it was studied the implementation of the JOSSO solution for Single Sign-On services. JOSSO was studied for its architecture and how it can be the mean that interconnects different technologies under its roof in order to provide high level services. All the components and various technologies are installed and configured in a user friendly interface console that is very easy to navigate. It comes in two different versions, one open source which is called JOSSO Community Edition and one commercial called JOSSO Enterprise Edition. Both editions were tested and analyzed in this thesis. Both have common features, but the Enterprise Edition has extra features for collaborating with commercial Cloud Providers and Authentication Providers. As a conclusion, JOSSO is one of the easiest to use and versatile SSO solution, introducing this kind of technology to users that do not have to be technologically skilled in a top level. It is a fine way for users to explore and learn about the SSO technology.
27. References

7. http://www.salesforce.com/eu/?ir=1