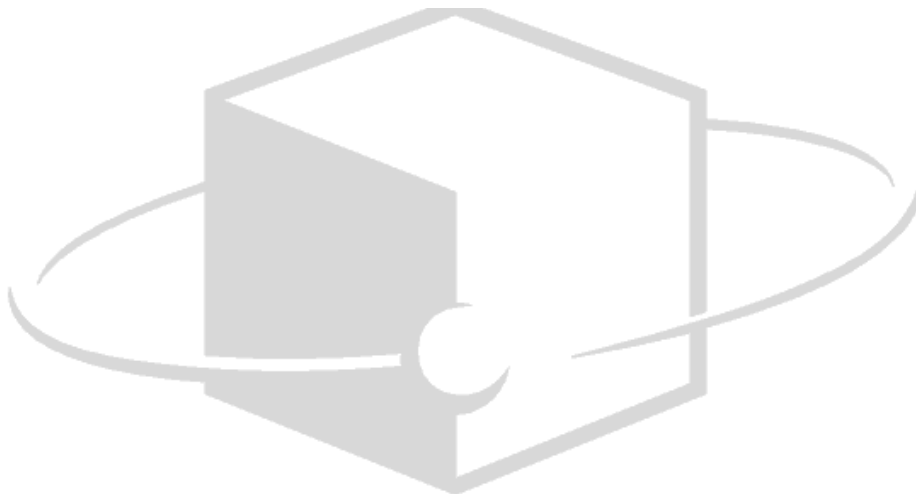

Installation Guide

instantOLAP

version 3.0

08/14/11



instantOLAP™

Copyright (C) 2002-2011 Thomas Behrends Softwareentwicklung e.K. All rights reserved.

This manual, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of such license. The content of this manual is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Thomas Behrends. Thomas Behrends assumes no responsibility or liability for any errors or inaccuracies that may appear in this book.

Except as permitted by such license, no part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of Thomas Behrends.

instantOLAP is a registered trademark of Thomas Behrends. Windows is a registered trademark of Microsoft Corporation. All other product and company names are trademarks or registered trademarks of their respective holders.

Typographical Conventions

Before you start using this guide, it is important to understand the terms and typographical conventions used in the documentation. The following kinds of formatting in the text identify special information.

Formatting convention	Type of Information
Special Bold	Items you must select, such as menu options, command buttons, or items in a list.
Brackets (<>)	Used for variable expressions such as parameters
Monospace font	Marks code examples or an expression-syntax
Emphasis	Use to emphasize the importance of a point
CAPITALS	Names of keys on the keyboard. for example, SHIFT, CTRL, or ALT
KEY+KEY	Key combinations for which the user must press and hold down one key and then press another, for example, CTRL+P, or ALT+F4

Content

Installation.....	6
Standard installation.....	6
Standard installation for Windows systems.....	6
Installation.....	6
instantOLAP on 64 bit systems.....	7
Setting up an instantOLAP service.....	7
Standard installation for Linux Systems.....	7
Custom installation in existing Application Servers.....	8
Deploying instantOLAP to an existing Tomcat server.....	8
Deploying war files.....	8
iolap.xml.....	9
iolapWS.xml.....	10
iolapWorkbench.xml.....	11
Additional libraries.....	11
Configuring the web applications.....	11
Adding data sources.....	11
Adding JDBC drivers.....	11
Data source definition.....	11
Updating from an earlier version.....	13
Replacing the war files.....	13
Updating related context configurations.....	13
Customizing.....	14
General settings.....	14
Enabling / disabling Guest mode.....	14
Changing the guest / default user.....	14
Setting the font folder for PDF / Word export.....	14
System tuning / memory usage.....	14
Changing the server port.....	14
Changing the context names.....	15
Using HTTPS.....	15
Enable / disable user automation.....	15

Changing the Workbench context.....	15
Repository settings.....	15
Changing the repository folder.....	16
Creating an SQL repository.....	16
Enabling version control.....	16
Data settings.....	16
Changing the data folder.....	16
Changing the compression level for dimensions and stores.....	17
Changing the read-buffer size.....	17
Changing the maximum number of open file handles.....	17
User management.....	17
Using LPAD servers.....	17
Using the Windows Name services.....	17
Automatic authentication with NTLM.....	17
Changing the role names.....	17
Mail settings.....	18
Creating a mail session.....	18
Configuring the mail session for the front and back-end.....	18
Cluster settings.....	19
Tomcat cluster setup.....	19
instantOLAP cluster extensions.....	19
Back-end configuration.....	19
Front-end configuration.....	20
Customizing the eventlog.....	20
Setting the maximum age for the eventlog.....	20
Creating an eventlog database.....	20
Using a CSV eventlog.....	21

Installation

instantOLAP is available as a standard installation package for Windows or Linux machines with included Application Server, Virtual Machine, demo data and anything else you need for a quick start. We recommend to use the standard installations for small or test installations.

You can also install instantOLAP in your own existing application server (Tomcat or other J2EE compatible server) or on other systems than Windows or Linux. We recommend this for advanced users only.

Standard installation

The standard installation is available for Windows or Linux systems. The standard installation includes the following components:

- Apache Tomcat Application Server (6.0.xx)
- Java Runtime Edition (1.6.xx)
- instantOLAP web archives for back-end and front-end server and the instantOLAP Workbench (administration tool)
- Default repository with demo reports and configuration
- HSQLDB demo database
- JDBC Drivers for HSQLDB, MS SQL and Sybase (jDts)
- Predefined HSQLDB user database with standard demo users (“admin”, “manager” and “guest”)
- The Windows installer also includes a desktop version of the Workbench and a getting started application with links for the most important programs and websites.

Both versions (Windows or Linux) are available on the instantOLAP download page <http://www.instantOLAP.com/downloads>.

Standard installation for Windows systems

Installation

The standard installation for Windows is a simple exe file. You can use the same installer for 32 or 64 bit systems. We recommend to run the installer as Administrator.

After you installed instantOLAP, a “Getting started” application will open. The application allows to start / stop the server in the console mode and to open the web front end and Workbench with a single mouse click.

The installer also creates a desktop icon for the “Getting started” application and for the desktop version of the Workbench. Additionally it creates a Windows start menu with extra items for installing the server as a Windows service.

instantOLAP on 64 bit Windows systems

The standard installation runs under 32bit and 64bit systems. For production systems we recommend to replace the JRE with the current 64 bit version – the 64 bit version allows to use more than 2 GB main memory for the instantOLAP process.

You can download the current JRE from <http://java.net>. The simplest way to replace the JRE is to unpack it into the instantOLAP home folder.

Setting up an instantOLAP service

On Windows servers we recommend to install instantOLAP as a service for production environments.

To install the service you can use the item “instantOLAP/Start/Service/Install Service” from the start menu or execute the batch file “<tomcat>/start/InstallService.bat” from the command line tool.

On 64 bit systems, you'll have to replace the JRE with a 64 bit version before running the service. You must also replace the file <instantOLAP>/tomcat/bin/tomcat6.exe with its 64 bit version tomcat6_64bit.exe (in the same folder).

To start or stop the service, you can use the items “Start Service” or “Stop Service” from the start menu. Alternatively you can use the batch files “StartService.bat” or “StopService.bat” from the command line tool or use the “Services” panel of the system settings.

Standard installation for Linux Systems

For Linux system instantOLAP is available as RPM package. To install the RPM package you must have the RPM manager installed on your system. Depending on your user account you may also need to execute the RPM with sudo:

```
sudo rpm -install instantOLAP_linux_x_y.rpm
```

By default, this will install instantOLAP in the folder “/opt/instantOLAP”.

To start the server, you must use the script “<instantOLAP>/tomcat/bin/startup.sh”. To stop it, use the script “<instantOLAP>/tomcat/bin/shutdown.sh”.

Custom installation in existing Application Servers

instantOLAP can also be deployed in other application servers than Tomcat or use existing Tomcat installations.

instantOLAP can also be downloaded as a package of different .war files (web archives), the standard format for J2EE web applications:

- iolap.war (front-end archive)
- iolapWS.war (back-end archive)
- iolapWorkbench.war (workbench archive)

However, after adding the war files to the application server, there is a number of settings you'll have to consider:

1. The application should have a configured user management. The configured user data source (e.g. a LDAP naming server) should contain all mandatory users and roles for the minimal instantOLAP installation.
2. Create a repository folder (the file system containing reports and configuration). The iolapWS context should be configured to link to this repository.
3. Create a data folder for all temporary files like dimensions or cubes. The iolapWS context must also be changed to find this data folder.

4. A mail service for report automation should be set up and configured in the iolap and iolapWS web context.

The following chapters explain most of this task for the Tomcat server. For other servers please refer to the server documentation.

Deploying instantOLAP to an existing Tomcat server

Deploying war files

There are several ways to deploy war files in a Tomcat server:

1. Simply drop the WAR file into the standard “webapps” folder of the server (usually <Tomcat>/webapps). This will force Tomcat to automatically install the applications. To configure the web applications, you must add context files to the folder – these files will contain your server specific settings for the war files.
2. Create context files in the configuration folder of your host, usually in the folder “<Tomcat>/conf/Catalina/localhost”. These context files must contain a link to the war files (anywhere on the server) and all server specific settings.

In both cases you need the context files, otherwise you can not configure the war files. We recommend to copy this standard context files into the folder “<tomcat>/conf/Catalina/localhost”:

iolap.xml

```
<Context path="/iolap" docBase="<path-to-wars>/iolap.war" debug="0" privileged="true" cookies="true" crossContext="true">

  <Valve
className="org.apache.catalina.authenticator.BasicAuthenticator" disableProxyCaching="false"/>

  <Environment name="webUrl"
    type="java.lang.String"
    value="http://localhost:8080/iolap"/>
  <Environment name="webserviceUrl"
    type="java.lang.String"
    value="http://localhost:8080/iolapWS"/>
  <Environment name="defaultUser"
    type="java.lang.String"
    value="guest"/>
  <Environment name="defaultPassword"
```

```

        type="java.lang.String"
        value="guest"/>
    <Environment name="enableGuestLogin"
        type="java.lang.Boolean"
        value="true"/>
    <Environment name="enableNTLMLogin"
        type="java.lang.Boolean"
        value="false"/>
    <Environment name="fontFolder"
        type="java.lang.String"
        value=""/>
    <Environment name="mailSessionName"
        type="java.lang.String"
        value="mailSession"/>
    <Environment name="mailFrom"
        type="java.lang.String"
        value=""/>
    <Environment name="workbenchContext"
        type="java.lang.String"
        value="/iolapWorkbench"/>

    <ResourceLink
        name="mailSession"
        global="mailSession"
        type="javax.mail.Session"/>
</Context>

```

iolapWS.xml

```

<Context path="/iolapWS" docBase="<path-to-
wars>/iolapWS.war" debug="0" privileged="true">
    <Environment name="repositoryPath"
        type="java.lang.String"
        value="../../repository"/>
    <Environment name="useVersions"
        type="java.lang.Boolean"
        value="false"/>
    <Environment name="dataPath"
        type="java.lang.String"
        value="../../data"/>
    <Environment name="userDataSource"
        type="java.lang.String"
        value="java:comp/env/jdbc/UserDS"/>
    <Environment name="clusterName"
        type="java.lang.String"
        value=""/>
    <Environment name="transactionTimeout"
        type="java.lang.Integer"
        value="120"/>
    <Environment name="tempIdleModelShutdownTime"
        type="java.lang.Integer"
        value="3600"/>

```

```
<Environment name="eventDataSourceName"
  type="java.lang.String"
  value="java:comp/env/EventDS"/>
<Environment name="maxEventAge"
  type="java.lang.Integer"
  value="168"/>
<Environment name="mailSessionName"
  type="java.lang.String"
  value=""/>
<Environment name="mailSystemFrom"
  type="java.lang.String"
  value="iolap@localhost"/>
<Environment name="mailSystemRecipient"
  type="java.lang.String"
  value="admin@localhost"/>

<Resource name="jdbc/EventDS"
  auth="Container"
  type="javax.sql.DataSource"
  driverClassName="org.hsqldb.jdbcDriver"
  url="jdbc:hsqldb:../../db/EVENTLOG"
  username="sa"
  password=""
  maxActive="1"/>

<ResourceLink name="jdbc/UserDS"
  global="jdbc/UserDS"
  type="javax.sql.DataSource"/>

</Context>
```

iolapWorkbench.xml

```
<Context path="/iolapWorkbench" docBase="<path-to-wars>/iolapWorkbench.war" debug="0" privileged="true">
</Context>
```

Additional libraries

Some libraries necessary for instantOLAP may miss in your Tomcat installation, particularly the JAR files for email automation (mail.jar and activation.jar).

These files are part of the standard instantOLAP installation. For a manual Tomcat installation, copy these files into the folder “<Tomcat>/lib”

Adding data sources

Adding JDBC drivers

For new target databases, you must add new JDBC drivers to the Tomcat server. JDBC drivers are usually JAR archives and available from the database vendors homepage.

The best place to add new driver jars is to copy them into the folder “<Tomcat>/lib”.

Data source definition

In production environments we recommend to set up data sources for you target databases and use them in your instantOLAP models. It is also possible to create a direct connection inside the configuration, but in this case different configurations do not share their connection pools.

To define a data source, declare it in the file “<tomcat>/conf/server.xml” or “<tomcat>/conf/localhost/Catalina/iolapWS.xml”. Data sources defined in the server.xml are globally visible, data source defined in iolapWS.xml are only visible inside the backend server (usually this is the easier and better way).

Defining a data-source in the back-end:

To add a new data source to the iolapWS.xml, simply append the following XML code to the end of the file:

```
<Resource
  name="<name>"
  auth="Container"
  type="javax.sql.DataSource"
  driverClassName="<driver>"
  url="<connection-URL>"
  username="<user-name>"
  password="<password>"/>
```

Replace all values <name>, <driver>, <connection-URL>, <user-name> and <password> with the correct values for your database. If you don't the driver class or connection-URL for your database, refer to the database documentation or use our JDBC connection HowTo <http://www.instantolap.com/howtos/jdbc-drivers>

This example only has the minimal number of settings for a data source. There are much more settings you can change, like the smallest or largest number of simultaneous connections. Please refer to the Tomcat data source documentation for a fully list of all available features.

Defining and importing a global data-source:

The syntax for global data-sources is equal to locally sources, but they must be defined in the file server.xml.

To use a global data-source in your back-end, add the following XML code to your backend configuration:

```
<ResourceLink name="<local name>"
              global="<global name>"
              type="javax.sql.DataSource"/>
```

Updating from an earlier version

To update from an earlier version of instantOLAP is usually an easy task, you will only have to replace the WAR files of your installation.

Replacing the war files

Download the newest WAR files and replace the previous files with the new versions. If the Tomcat server is configured to unpack the WAR files in its webapp folder, you must also remove the unpacked folders from the folder.

You should never remove the unpacked folders from the webapp folder while the Tomcat server is running – this will also destroy all configuration files (iolap.xml and iolapWS.xml) of your WAR files!

We strongly recommend to configure Tomcat not to unpack WAR files in its webapp folder. You can change this setting with the “unpackWars” property of the host node in the server.xml file.

Updating related context configurations

Most time you will not need to update the context configurations (iolap.xml and iolapWS.xml) unless you want to add a new property which wasn't available in the earlier version.

All properties have default values defined in the file WEB-INF/web.xml in the WAR archives. Therefore, all new version will also work without updating the configuration files and use their default values for new properties.

Customizing

General settings

Enabling / disabling Guest mode

In the standard installation, instantOLAP uses the Guest by default. To disable it, change the property “enableGuestMode” in the back-end settings (iolapWS.xml) to “false”.

Note that the guest user is still mandatory, even with disabled guest-mode. This is because the front-end still communications with the back-end using the guest account until a user logs in.

Changing the guest / default user

The default guest user is “guest” (password guest). If you want to change the name or password of the standard user, you must also enter the new name and / or password in the front-end context.

To do this, change the properties “guestUser” and “guestPassword” in the iolap settings (iolap.xml).

Setting the font folder for PDF / Word export

For PDF and Word-Export the server needs access to a folder with all necessary true type fonts. Usually instantOLAP will find the standard font-folder of your machine automatically, but if there are problems you can configure the folder manually.

Define the folder location with the property “fontFolder” of the front-end configuration (iolap.xml).

System tuning / memory usage

The memory usage is usually managed automatically by the server. There is only a single property which allows to limit the memory consumption for a report: “maxMemoryNodes”.

This property determines the largest size of a result in the system memory. Any report with more data will start swapping its result to the hard disk of the server.

The default value for this property is “100000000” - you can change this setting to a smaller amount when you experience a Heap Space Error while executing a report.

Changing the server port

The standard installation of instantOLAP runs on port 8080. To change this, simply change the port number in the Tomcat configuration (<Tomcat>/configuration/server.xml).

After you changed the port, you must also tell the front-end application where to find the back-end and itself. The front-end configuration (iolap.xml) has two properties “webUrl” and “backendUrl”, which also contain the port number. Change these URLs after you changed the server port.

Changing the context names

Like changing the port, it is also possible to change the context names of the application. The standard context name is “/iolap” for the front-end, “/iolapWS” for the back-end and “/iolapWorkbench” for the Workbench.

You can change the context names in the configuration files for front-end (iolap.xml), back-end (iolapWS.xml) or Workbench (iolapWorkbench.xml) with the attribute “path” of their root nodes.

Like with the port number, you must also edit the “backendUrl” and “webUrl” in the front-end configuration after you changed a context path. If you changed the Workbench path, you should edit the property “workbenchContext” of the front-end.

Changing the Workbench context

The front-end offers a link to the Workbench web-application for administrators. If you changed the context path of the Workbench, you should also edit the “workbenchContext” property of the front-end, otherwise this link does not work.

Enable / disable user automation

User automation allows all “power users” to create and edit their own automation tasks (like sending reports as email or exporting them to the server file system).

By setting the front-end property “enableUserAutomation” to false, only administrators will be able to create automations.

Repository settings

The repository is a file system, that holds all project files for instantOLAP, including all reports and model configurations.

Usually the repository is a simple folder on the server but it can also store it data on a SQL server.

Changing the repository folder

To change to folder of the repository simply change the property “repositoryPath” of the back-end configuration (iolapWS.xml).

If the front-end runs on the same Server, you don't need to set the repository for the front-end server (it only uses the repository to load data for maps). If the front-end runs on a separate server you must edit the property “repositoryPath” of its configuration (iolap.xml).

Creating an SQL repository

Instead of using a file-system the repository can also store its content on a SQL database. For this you must create a database with the following table:

```
CREATE TABLE REPOSITORY(FOLDER VARCHAR(255), PATH
VARCHAR(255), ENTRYTYPE CHAR(1), ENTRYTIMESTAMP LONG,
ENTRYDATA BLOB, ENTRYLOCK VARCHAR(128))
```

When the repository table exists, add a data-source for the data-base, either in the Tomcat configuration (server.xml) or backend configuration (iolapWS.xml). Then edit the back-end property “repositoryDataSource” and enter the JDNI name of the data source.

Enabling version control

The repository uses a very simple version control system which only keeps all versions of any changed file in a “.version” sub-folder. The Workbench has no version control function and older versions can only be restored manually.

To enable the version keeping, editor the property “keepVersions” of the back-end configuration and set it to “true”.

Data settings

The data folder holds all temporary files like dimension files, caches or stores. This folder has a lot of I/O traffic. Therefore we recommend to put it on a fast hard drive.

Altering the data folder

Change the folder with the property “dataFolder” in the back-end settings (iolapWS.xml). The folder must only exist, the server manages anything else (like sub-folders) automatically.

In a cluster, all back-end servers must use the same data folder. If they use different data folders they are not able to share their dimensions and stores and will query unnecessary data from the database servers.

Changing the compression level for dimensions and stores

instantOLAP is able to compress all data files. This will shrink all files but may result in longer build times for dimensions or stores. But depending on the server or storage system, the build time for compressed files could also be faster than for uncompressed.

To change the compression level from 0 (the default value), edit the property “compressionLevel” of the back-end settings (iolapWS.xml) and enter a value between 0 (no compression) and 9 (max).

Changing the read-buffer size

For dimension files and stores, instantOLAP uses read-buffers to increase their performance. The default size for this buffers is 128 pages (each page has 4kb). You can increase this size by changing the “readBufferSize” property of the back-end (iolapWS.xml), but consider that this will also increase the overall memory consumption of your system,

Changing the maximum number of open file handles

On Linux systems, the maximum number of open file handles is usually 1024. On instantOLAP installations with a large number of dimensions (>500), increase this with the “ulimit” command.

User management

User authentication in Tomcat uses “Realms”. A Realm is a kind of data-source for authentication and checks user-name / password combinations and delivers all roles assigned to a user.

Grouped JDBC Realm

The default installation of instantOLAP uses the GroupedJdbcRealm (a special realm delivered with instantOLAP). This realm holds all user data in a JDBC data-source and organizes all users in a hierarchical group structure.

We recommend this realm for instantOLAP installations unless you want to use an existing other data-source like JNDI servers or Windows domains.

Only with this realm your users will be able to edit users, roles and groups with the instantOLAP Workbench. With all other realms, user management is disabled in the Workbench.

Creating a database for Grouped JDBC Realms

When installing instantOLAP manually or in productive systems with large user numbers, we recommend to create a user database on a database server. The standard installation only uses a file bases HSQLDB database.

For the database you must create the following five tables:

```
CREATE TABLE ROLES (ROLENAME VARCHAR(255), ROLECOMMENT  
VARCHAR(255))
```

```
CREATE TABLE USERS (USERNAME VARCHAR(255), GROUPNAME  
VARCHAR(255), USERCOMMENT VARCHAR(255), PASSWORD  
VARCHAR(255))
```

```
CREATE TABLE USERROLES (USERNAME VARCHAR(255), ROLENAME  
VARCHAR(255))
```

```
CREATE TABLE GROUPS (GROUPNAME  
VARCHAR(255), PARENTGROUPNAME VARCHAR(255), GROUPCOMMENT  
VARCHAR(255))
```

```
CREATE TABLE GROUPROLES (GROUPNAME VARCHAR(255), ROLENAME  
VARCHAR(255))
```

After creating the tables add a data-source for the database into your server.xml:

```
<Resource  
  name="jdbc/UserDS"
```

```

auth="Container"
type="javax.sql.DataSource"
driverClassName="org.hsqldb.jdbcDriver"
url="jdbc:hsqldb:../../db/USERDB"
username="sa"
password=""/>

```

Setting up the realm

Now you can use the new data-source for your Grouped JDBC realm. Add the following xml code to your server.xml (and replace the existing realm):

```

<Realm
  className="de.instantolap.realm.GroupedDataSourceRealm"
  dataSourceName="jdbc/UserDS"
  cachetime="30"/>

```

Using LPAD servers

Use Tomcats JndiRealm to connect LDAP servers. Because the JndiRealm has many properties, its document would be too large for this document. Please refer to the realm documentation at:

<http://tomcat.apache.org/tomcat-6.0-doc/realm-howto.html#JNDIRealm>;

Using the Windows Name services

Windows Name Service is a special case: Instead of connecting a Name Service with the JndiRealm (which is also possible), the standard installation also comes with a very simple native connector for Windows domains.

This connector uses a native library and only works on Windows installations. To activate the Windows realm, insert the following realm into the server.xml:

```

<realm class="de.instantolap.realm.WindowsRealm"
  userRole="<group>" powerUserRole="<group>"
  adminRole="<group>" managerRole="<group>"/>

```

The basic roles must be mapped to Windows rules with the four attributes "userRole", "powerRole", "adminRole" and "managerRole". Input Windows group names with their domain or "PREDEFINED\" or (VORDEFINIERT\ for German servers) here:

Basic Role	Attribute	Example
------------	-----------	---------

lolapUser	UserRole	PREDEFINED\User
lolapPowerUser	PowerUser	PREDEFINED\User
lolapAdministrator	AdministratorRole	PREDEFINED\Administrator
lolapManager	ManagerRole	PREDEFINED\Administrator

All groups from the Windows domain become visible in the Java sessions. Every group is visible with or without their domain.

Automatic authentication with NTLM

With the WindowRealm you can also activate automatic NTLM login for your server. For this perform the following two steps:

1. Set the property "ntlmLogin" in the front-end configuration to "true"
2. Patch the file WEB-INF/web.xml in the WAR archive lolap.war and uncomment the "NTLM SSO" section. Here you must also insert all role mappings for the basic roles like in the section before.

Changing the role names

Mail settings

instantOLAP uses mail sessions for automated reports and error notifications. For both the Tomcat server must be connected to an existing SMTP server, otherwise it can not send mails.

Creating a mail session

The standard installation already contains a configuration for the mail session in its server.xml file, but you must change it to your real SMTP settings to use it.

If there is no mail-session, enter the following code to the "GlobalNamingResources" node of your server.xml

```
<Resource
    name="mailSession"
    auth="Container"
```

```
type="javax.mail.Session"  
mail.smtp.host="<server>"/>
```

Replace the server address with the name or IP of your SMTP server. This XML code is only the minimal definition for a mail session, refer to the Tomcat documentation for more properties like passwords or encryption.

Note that your server needs the libraries `activation.jar` and `mail.jar` to send mails. These files are part of the standard distribution but may miss in other Tomcat installations.

Configuring the mail session for the front and back-end

When the mail session exists, you can refer to it in the front-end configuration (`iolap.xml`) or back-end configuration (`iolapWS.xml`):

- The property “mailSessionName” holds the JDNI name of the mail session resource. The default value for this property “mailSession”, equal to the name of the initial configuration in the `server.xml`. You don't have to change this property in a standard installation.
- The mail session must be imported into the `iolap` context. Use a `ResourceLink` entry to make it visible for your context:

```
<ResourceLink                                name="mailSession"  
global="mailSession"    type="javax.mail.Session"/>
```

The standard installation already contains this entry.

After the mail session is visible for the front-end or back-end, you can use the specific properties of both context configurations.

The following front-end properties exist:

- The “mailFrom” property allows to change the mail address all automations will use as sender.

The following back-end properties exist:

- The property “mailSystemRecipient” allows to specify one or more (comma separated) email addresses for administrators. The system will send all interesting system events to these addresses.

- The property “mailSystemFrom” specifies the mail address these mails use as sender.

Server names

When using more than one instantOLAP servers, you should give them unique server names. Change both configurations (iolap.xml and iolapWS.xml) and enter a unique server name (property “serverName”) here.

The server names are used for logging (in the eventlog) and optionally for automation settings (you can limit certain automations to run only on a certain server).

Cluster settings

instantOLAP front- and back-ends are able to run in a cluster configuration. Basically it uses the standard cluster techniques of Tomcat but needs some extra settings, because back-end server can exchange data.

Tomcat cluster setup

Tomcat servers are able to run in a cluster. When clustering application servers, they continuously exchange all session information and are able to continue a user session when one of the server is no longer available.

There are different ways to run Tomcat in cluster mode. The easiest way is to use a SimpleTcpCluster. You only need to uncomment the cluster settings in the server.xml file to run your servers in clustered mode.

instantOLAP cluster extensions

Back-end configuration

Back-end servers exchange some information, especially instance numbers. To find a valid instance number, each back-end server should know to which cluster it belongs. Unless you run more than one cluster in your network there is no need to change anything, but with more than one you must change the “clusterName” property of the back-end servers (all back-end servers of the same cluster should use the same cluster name).

Also, all back-end servers of one cluster must use the same repository and data folder – even if they run on different machines, they should use a shared network folder. If the servers don't share their data folders, they will perform unnecessary queries on your databases.

Front-end configuration

If you own more than one back-end you must tell the front-ends to use all servers and where to find them. There are different ways to use multiple back-ends:

- You can use an extra Web-Server like Apache 2.0 and map multiple back-end servers to a context – then the Apache server will diverse all requests to all available back-ends. In this case, the “backendUrl” of your front-end must point to the new Web-Server.
- You can specify more than one URL the “backendUrl” property of your front-end and separate them with commas. In this case the front-end will diverse all request with a round-robin method between all available back-ends.

This is only the scenario for using multiple back-ends with a single front-end server. If you also want to use multiple front-ends, you will need to set up an extra Web-Server like Apache and to map all front-ends to one URL.

Customizing the eventlog

Every event in instantOLAP is protooled in the eventlog. Events are errors, the system start or shutdown, re-buildings of dimensions or stores and anything else. The eventlog is useful for debugging or performance analysis of your server.

Setting the maximum age for the eventlog

Because the eventlog can grow very fast, it only holds the data for the last x days (the default value is 7 days). To change the maximum age for events in the log, change the property “maxEventAge” of the back-end server. Enter the age in hours here (the default value is 186).

Creating an eventlog database

The standard installation uses a HSQLDB (file database) for the eventlog. For larger installations or longer event ages than 7 days, we recommend to migrate the database to a real database server.

The database must contain the following two tables:

```
CREATE TABLE EVENTLOG (SERVERNAME VARCHAR(64), EVENTTIME
CHAR(18), EVENTNUM INTEGER, SESSIONID
VARCHAR(64), USERNAME VARCHAR(64), MODELNAME
VARCHAR(256), QUERYNAME VARCHAR(255), EVENTTEXT
VARCHAR(255), DURATION BIGINT, ERRORMESSAGE VARCHAR(255))

CREATE TABLE SYSTEMLOG (SERVERNAME VARCHAR(64), EVENTTIME
CHAR(18), FREEMEMORY BIGINT, MAXMEMORY BIGINT, TOTALMEMORY
BIGINT, SESSIONCOUNT INTEGER)

CREATE TABLE EVENTNAME (EVENT INTEGER, EVENTNAME
VARCHAR(255))
```

To use your new eventlog database, change the eventlog data-source defined in the back-end configuration to the new driver and connection URL.

Using a CSV eventlog

It is also possible to store all events in a CSV file. All events will be appended to the file and the file will always grow unless you delete its content.

To activate the CSV log enter the filename of the target file in the property “eventPath” of the back-end configuration.