



Light RMA Technical Guide Book

© Intalio
Pierre Pavageau

Contents

Useful Terms	3
Installing the RMA.....	4
Purpose	4
Components.....	4
Installing the light RMA.....	5
Installing and running MySQL and activeMQ.....	5
Installing Intalio Enterprise Edition Server.....	5
Actions to perform in MySQL	5
Actions to perform in activeMQ.....	6
Files that need to be added/deleted/replaced	6
Files that need to be edited	7
Further RMA configuration: tempo and instance cleanup	9
Configuring Tempo	9
Configuring the cron-based process instance cleanup.....	9
Changing the main ports used by Intalio.....	9
Configuring and installing the processes	10
Setting up Intalio Designer	10
Configuring database access for Intalio Designer.....	11
Configuring the processes	12
Configuring the database connectors.....	13
Configuring the form endpoint.....	14
Configuring the WSDLs	14
Configuring the deployment.....	15
Configuring the archived deployable projects.....	16
Deploying the processes	18
Running the RMA	18
Troubleshooting.....	19
Appendix: cron-based instance cleanup.....	19
Process cron configuration.....	21
Cron expression.....	22
Cleanup configuration filters	22
Troubleshooting	24
Lock time-outs and/or deadlocks in the dbms	24
JTA Transaction time-outs	24
Cleaning up cannot keep up with generation of data	24

Useful Terms

RMA: Resource Management Application

UI-FW: User Interface Framework, it is the main UI component of the Intalio runtime server. It has been heavily customized for this project.

TMS: Task Management Service, designates a web-service present in the Intalio BPMS server out of the box. It has been heavily customized for this project

FMR: Flight Movement Record. This is data that comes from Compass into the RMA via web-service.

TA: Abbreviation for Turnaround Activity. It can be assimilated to a task in Intalio

Installing the RMA

Purpose

This section describes how to install the light RMA. The light RMA consists of a modified version of the Intalio 6.0 Enterprise Edition server with some processes and services developed by Intalio process experts, namely Pierre Pavageau and Ihab El-Alami. This guide uses version 6.0.0.094 of Intalio Enterprise server as reference, therefore some information might change if using a different version (line numbers in text files, for example).

Components

- The latest version of the Sun Java JDK 1.6 (tested with version 1.6.0_13)
- Intalio Enterprise Edition runtime (server), version 6.0.0.094 or later. You may download this from the Intalio support website: <http://support.intalio.com>
- ActiveMQ 5.2.0. ActiveMQ is a full JMS message broker, provided by the Apache foundation free of charge. It can be downloaded at: <http://activemq.apache.org/download.html>
- MySQL 5.1. MySQL is a full-fledged database that the Intalio Enterprise Edition server uses as a back-end. It can be downloaded at: <http://dev.mysql.com/downloads/mysql/5.1.html>
- MySQL JDBC connector JAR version 5.1.8. It can be downloaded at: <http://dev.mysql.com/downloads/connector/j/5.1.html>
- Intalio Enterprise subscription license. This file must be obtained by SITA by downloading it from the Intalio support website
- Custom ui-fw.war. UI-FW is the user interface module of the Intalio BPMS runtime, and has been deeply customized to match the requirements of the RMA.
- Custom TMS. TMS (Task Management Service) is a web-service that is natively part of the Intalio BPMS runtime. It has been heavily customized to match the requirements of the RMA
- Custom web-service developed by Intalio for this project. This service was developed by Pierre Pavageau and Ihab El-Alami
- Custom properties file: sita.properties. This file contains properties that are exclusive to this project. The custom web-service mentioned above uses this properties file
- SQL script containing the information needed for the RMA database, such as the list of mechanics/avionics/coordinators (sita.sql)
- Custom security.xml file. Contains all the users of the RMA (authentication system). In the long run, this system should be replaced with an integration with an LDAP directory
- Custom processes, to be deployed on the Intalio server. These processes were developed by Pierre Pavageau and Ihab El-Alami
- Left.gif and Right.gif, two icons created by Intalio and which are used in the RMA GUI
- SQL script containing the information needed for the FMR logging database (log.sql)

Installing the light RMA

Installing and running MySQL and activeMQ

This guide doesn't contain a detailed procedure for installing and running MySQL and activeMQ. Please follow the installation instructions on the respective websites of these components. Both of these components need to be running before starting the Intalio server. The RMA cannot work if one or both of these components is inactive.

Installing Intalio Enterprise Edition Server

After downloading the Intalio Enterprise Edition server (version 6.0.0.094 or later, earlier versions will NOT work), you must unzip the contents of the zip archive on your machine, to a location whose path contains no space. This is a very important point. In the rest of this guide, the root path of the Intalio server will be marked as **<intalio>**.

For example, at the time that this guide is written, in Lisbon the root directory of the Intalio server is /opt/intalio5. Therefore, for this example, when mentioning <intalio>/var/config/schedules.xml, it actually refers to /opt/intalio5/var/config/schedules.xml

Once unzipped and before starting the Intalio server, several things need to be done, executing them in the order in which they are described below is preferable:

Actions to perform in MySQL

Before you do anything in MySQL, open **<intalio>/databases/MySQL/BPMS.sql**. In this file, at line 283, there is a statement that starts with *create table tempo_pa...* You must comment this statement out (put */** without the quotes at the beginning of the line, and **/* without the quotes at the end of the line). This table will be created by the application itself later on, with specific options that cannot be scripted. As an alternative, you can replace the existing BPMS.sql script with the one provided by Intalio. In that case, please make sure that the files match with the exception of the line.

You need to create at least two databases, or three if you prefer. One is the main back-end database for the Intalio server, the other one is the FMR logging database and the last one is where you could store application data such as shifts, coordinators, mechanics, and avionics. If you want, you can store this application data in the same database as the Intalio back-end database (which is the current situation in the Lisbon environment), or in the logging database. In that case you only need to create two different databases.

The next thing to do is to run the BPMS.sql script against the database corresponding to the Intalio server back-end database. This database will be called **BPMSDB** in the rest of this guide. Then you

need to run the sita.sql script against the database corresponding to the application data. This database will be called **APPDATA** in the rest of this guide. Finally, you need to run the log.sql script against the logging database. This database will be called **FMRLOG** in the rest of this guide. BPM SDB and FMRLOG need to be distinct, but APPDATA can be merged in any of the other two, or independent.

You also need to change the value of the *max_allowed_packet* variable in MySQL to something larger than the default value, for instance 1073741824, which corresponds to 1024*1024*1024. This is needed as the Intalio server will be making large queries, and the default value can easily be too small to support them.

Actions to perform in activeMQ

Install and start activeMQ. If downloaded from the website mentioned earlier, installing it consists of unzipping the archive to your machine. Once activeMQ is started, you must create three queues. The names of these queues is very important: **so.tasks.queue** , **co.task.queue** , **so.msg.queue**

You can access the activeMQ GUI via any web browser at the address <http://<host>:8161/admin>, where you need to replace <host> by the host address where activeMQ has been installed. For example, if you are accessing activeMQ from the same machine as the one it is running on, then you need to replace <host> by "localhost" without quotes.

Files that need to be added/deleted/replaced

- The SITA license file must be placed in <intalio>/var/config
- The SITA custom web-service must be placed in <intalio>/webapps/axis2/WEB-INF/services. Please note that the name of the file is not important as long as the extension is .aar
- In <intalio>/webapps/axis2/WEB-INF/services, the default TMS (tempo-tms-service-XXX.aar) must be **replaced** by the custom TMS service. Please note that the name of the file is not important as long as the extension is .aar, but it is important that both TMS implementations **don't coexist**
- The sita.properties file must be placed in <intalio>/conf
- In <intalio>/webapps, you must delete the **ui-fw** directory and all its content, and place the custom ui-fw.war file there (in <intalio>/webapps). Please note that the name of the file is very important, it must be called ui-fw.war all in lower case. If it is named differently, you must rename it
- In <intalio>/var/deploy, delete everything, all directories and all files if any. The <intalio>/var/deploy directory must be empty. You will need to place some files here later, but they require some edition first. This will be detailed further in this guide
- Left.gif and Right.gif both need to be placed in <intalio>/webapps/gi/files/JSX/images/list
- In <intalio>/common/lib, you must delete the existing MySQL JDBC connector JAR (version 5.0.4) and replace it with version 5.1.8
- In <intalio>/var/config, you must **replace** the existing security.xml file with the custom one

Files that need to be edited

Once you have done all the steps mentioned above, several files need to be edited in different places on the server. This section will describe these different editions and provide some insight on what these editions do.

<intalio>/bin/setenv.sh or **<intalio>/bin/setenv.bat**, depending on your operating system: there are two things to do in this file. The first is to change the value of the maximum allowed memory (-Xmx in the file), and set the value to 2048m instead of the default value of 512m. The machine you are running on needs to have at least 2GB of RAM to allow this. You also need to add the following property:

-Dorg.apache.ode.processInstanceDeletion.transactionSize=300

This property determines the size of the transaction when deleting the completed process instances (automatic cron procedure).

<intalio>/conf/resources.properties, this is the file where the back-end database for the Intalio server is defined. There are several things to be changed in this file. All the following modifications concern existing properties, no new property need to be added:

- The maxPoolSize property must be set to **150**
- The driverClassName property must be changed to *"com.mysql.jdbc.Driver"* without the quotes
- The url property needs to be changed to *"jdbc:mysql://localhost:3306/<databaseName>"* without the quotes, where *<databaseName>* needs to be replaced by the name of the **BPMSDB** database. This assumes that the Intalio server and MySQL are running on the same machine, otherwise you need to change "localhost" to the name of the host where MySQL is running on
- The user property needs to be set with the name of a configured user in MySQL who has all permissions on the BPMSDB database (you can use *root*)
- The password property needs to be set with the password for that user

<intalio>/conf/sita.properties, this is the file where custom properties for the RMA are stored. There are 4 custom properties in this file:

- dburl: this property specifies the location of the **APPDATA** database. The value of this property should be *"jdbc:mysql://localhost:3306/<databaseName>"* without the quotes, where *<databaseName>* needs to be replaced by the name of the APPDATA database. This assumes that the Intalio server and MySQL are running on the same machine, otherwise you need to change "localhost" to the name of the host where MySQL is running on
- dbuser: this property specifies the MySQL user for the APPDATA database
- dbpassword: this property specifies the password for that user
- fmrlog: this property is a switch that allows to turn FMR logging on or off. It should only take one of two possible values: **ON** and **OFF**. Any other value will be considered as OFF. When FMR logging isn't needed, it should be kept off

<intalio>/conf/server.xml, this is the main configuration file for the server. In it you need to configure the ports that you want the Intalio server to use for its communications. The lines that need to be edited, if you want to change the different ports are: 30, 76, 110, 112, and 136. Currently, the main port configured in the Lisbon environment is 9080 (line 110), and 9443 for redirection (lines 76 and 112). Please note that later in this guide, there is a procedure that allows you to automatically change the ports for the entire Intalio application server. That procedure disregards this specific file (**<intalio>/conf/server.xml**), therefore this is the only file that which needs to be configured by hand for port configuration.

<intalio>/conf/btm-config.properties, you need to add a new property in this file:

bitronix.tm.timer.defaultTransactionTimeout=300

This property defines the duration after which a transaction throws a timeout error. Given the amount of data that the RMA handles, transactions can sometimes take a relatively long time, especially regarding process instance cleanup, which is an automated procedure. A value of 300 works well.

<intalio>/var/config/tempo-tms.xml, line 46, there is an XML statement that is commented out by default. You must uncomment it. The statement is the following:
`<entry><key><value>openjpa.jdbc.SynchronizeMappings</value></key><value>buildSchema</value></entry>`. This statement must be uncommented.

<intalio>/var/config/ode-axis2.properties, in this file, there is a property entitled ode-axis2.threads.pool.size. You must change its value to **20**.

<intalio>/var/log/log4j.properties, this file contains the Intalio server log configuration. This is unrelated to the FMRLOG database, which is custom to this project. In it, there is a property entitled *log4j.appender.CONSOLE.Threshold*. You must change the value of this property to “OFF” without the quotes. Otherwise, certain log files will grow constantly in size if there are things to be logged, which is generally the case.

<intalio>/webapps/ode/WEB-INF/conf/axis2.xml, this is the main configuration file for ODE (Orchestration Director Engine), which is the engine that powers the Intalio application server. In this file, you need to enable JMS communication. To do so, there are two XML blocks that you must uncomment (they are commented by default). From line 137 to 155, you have the configuration for JMS listeners. The entire block must be uncommented. If you are using the default activeMQ configuration, you don’t need to make any change to the default values in this section, otherwise you need to adapt to your configuration. Line 194, you have the configuration for JMS senders. You must uncomment the following XML statement: `<transportSender name="jms" class="org.apache.axis2.transport.jms.JMSSender"/>`

You may also want to configure the cron-based process instance data cleanup as well as some parameters in ui-fw. Nevertheless, these aren’t required steps in order to get the system running, therefore they will be detailed in a later part of this guide.

Further RMA configuration: tempo and instance cleanup

There are two custom configurations that need to be done for in the RMA for it to work properly. These aren't necessary to run Intalio, but in order to run RMA properly given the Lisbon environment (TAP) specifications, certain configurations need to be done.

Configuring Tempo

In the Intalio server file system, open `<intalio>/var/config/tempo-ui-fw.xml`. In here, there are several properties that you can change:

- refreshTime: this value defines the time duration in seconds between two refreshes of the UI
- sessionTimeout: this value defines the duration in minutes when a user session will automatically finish if a user is idle. It should be set to **510** (8 hours and a half, the duration of a shift) for the TAP RMA
- You must comment out the entire line for the claimTaskOnOpen property. This property is not yet functional

Configuring the cron-based process instance cleanup

In the server file system, open `<intalio>/var/config/schedules.xml`. By default, there is a commented out section. You must replace that section with the following:

```
<sched:schedule when="*/6 * * * * ?">
  <sched:cleanup>
    <dd:filter><![CDATA[last_active<-1m status=completed]]></dd:filter>
  </sched:cleanup>
</sched:schedule>
```

This configuration means that all process instances with the *completed* status and older than one minute (at the time the cron procedure is run) will be deleted along with their respective instance data. This does not influence task data, as in this project, tasks don't belong to any process properly speaking. This cron-procedure will try to start every ten seconds. Once it is running, it will not try to run again until the first procedure is done.

You will find a complete explanation on how to configure this cron-based cleanup at the end of this guide.

Changing the main ports used by Intalio

By default, Intalio uses port 8080 for its communications. In the Lisbon environment, this port has been changed to 9080. There are a lot of different places where this port is mentioned, and changing them all by hand would be very tedious, therefore Intalio provides a tool that allows users to change the ports used by the application easily. Open `<intalio>/extras`, you will see several files.

In order to change the main http port used by Intalio you need to run one of the following commands, depending on your operating system:

```
groovy.bat change_http_port.groovy OR groovy.sh change_http_port.groovy
```

Then follow the instructions in the terminal. You will be prompted to choose between http and https, you can configure both if you want, just run the script twice. Then you will be asked for the port number you want to use.

You can also change the RMI port used by running the *change_rmi_port.groovy* with an analog command. By default it is 2099, it hasn't been changed as part of this project.

WARNING: this procedure does NOT edit `<intalio>/conf/server.xml`. This specific file needs to be edited manually, as mentioned previously in this guide.

Configuring and installing the processes

Intalio provides ready-to-deploy versions of the processes for the Lisbon environment. That means that they are configured as follows:

- The main port used by the Intalio server is 9080
- BPMSDB and APPDATA are both the same database called tap
- The FMRLOG database is called fmrlog
- The IP address of the server on which Intalio is running is 172.17.128.120
- The Intalio server and activeMQ are both running on the same machine
- All database access is done with a single user: username=root, password=appear. This user has all permissions on all tables in all databases in MySQL

Therefore, in the Lisbon environment, all you need to do to install the processes on the intalio server is unzip the corresponding archive in `<intalio>/var/deploy`. This should create two directories, TaskManager and SITA.

WARNING: Make sure you don't confuse the designer project archive with the deployment archive. They contain very different files, and mixing them up will not work at all.

Nevertheless, the deployment archive will be difficult to use in any other environment, therefore this section of the guide is meant to describe how to configure and deploy manually the processes. You will need Intalio Enterprise Edition Designer, that you can download from the Intalio support website (<http://support.intalio.com>), version 6.0.0.098 or later. Previous versions will NOT work. Intalio Designer does not need to be installed on the same machine as the Intalio server, they are two separate components and can be used on different machines.

Setting up Intalio Designer

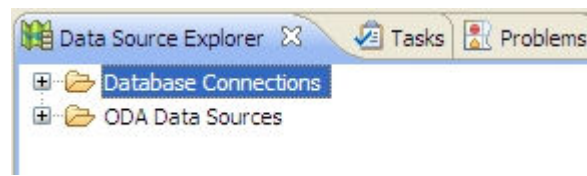
Once you have downloaded and installed Intalio Designer (version 6.0.0.098 or later), you must place the SITA license in the Designer installation directory (the same directory where you can find designer.exe, on Windows platforms). Then you must place left.gif and right.gif in `<designer>/plugins/com.intalio.bpms.designer.gi.builder_xxx/intalioajax/JSX/images/list`. Finally, you

can start the program by launching designer.exe (on Windows platforms). You must have also the MySQL JDBC connector Jar (version 5.1.8) somewhere on your machine. The first thing you will need to do is choose a location for your designer workspace. This is where your designer configuration will be stored as well as all your projects. You can have several workspaces if you like, with different configurations and different projects. Once you have chosen a workspace, the workbench will open.


Configuring database access for Intalio Designer

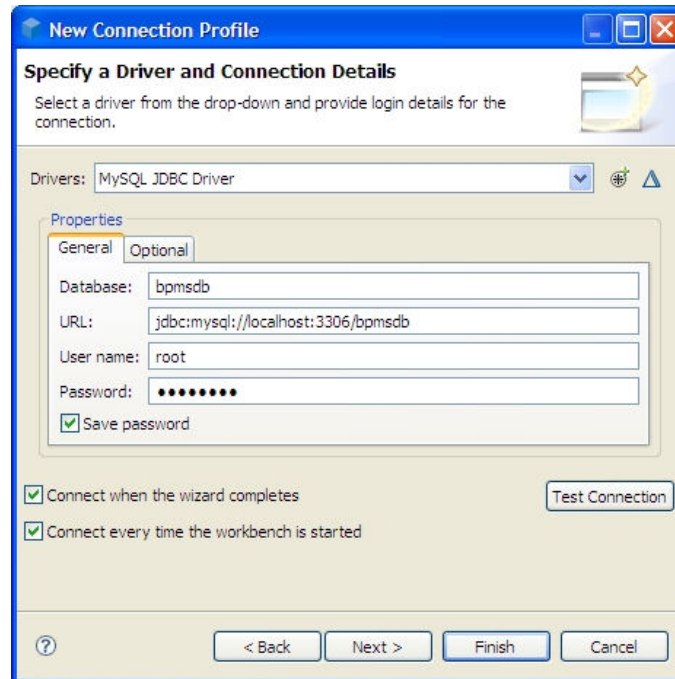
The Intalio database connector generates XML schemas corresponding to the input/output of the SQL queries that you want to use as part of business processes. Therefore, Intalio Designer needs to have access to the database(s) that the processes will communicate with in order to introspect them and be able to generate those schemas. The first thing to do is to make sure that you have the MySQL JDBC Driver somewhere on the machine on which you are running Intalio Designer.

You then need to open the **Data Source Explorer** view in Designer. This view is closed by default, so you need to open it. In the top menu bar, select *window -> show view -> other*. In the popup window that comes up, select *Data Management -> Data Source Explorer*. This should open a view that looks like this:



Right-click on *Database Connections* and select *new*. In the window that pops up, select *MySQL* and give your connection a name. Since you will need two or three different connections (one for BPMMSDB, one for APPDATA and one for FMRLOG), it is a good idea to name your database connections accordingly. After that, click the *Next* button.

In the next window, the first thing you will need to do is select your JDBC driver. If you try to unfold the drop-down menu, you will notice that it is empty, there are no drivers. You need to click on the button right next to the drop down menu: . This will allow you to set a location for your JDBC driver, which will become part of your designer configuration. In the window that pops up, select *MySQL JDBC Driver 5.1*, then go to the *Jar list* tab. In this tab, select the driver if there is any, and click on the *Edit JAR/Zip* button. This will allow you to browse to the location of the MySQL JDBC Driver (version 5.1.8) on your machine. Select it, then you should be able to click on the *Ok* button in that window, which should take you back to the connection configuration. This time, you will be able to select your newly configured driver in the drop-down menu, and configure your database connection. Below is an example of what this configuration should look like:



Once you have done this configuration, you can check that the connection works by clicking on the *Test Connection* button. You should get a “*Ping succeeded*” message when you do. If you don’t get this message, that means there is a problem with your database connection or configuration. Once the ping has succeeded, you can click on the *Finish* button.

You need to create as many database connections as the number of different databases that you will be connecting to. For this project, there need to be at least two (reminder: BPMSDB and FMRLOG need to be distinct).

In the *Data Source Explorer* view, you should now see your database(s) and be able to navigate through the tables they contain. Intalio Designer is now configured to connect to the different databases that your processes will be using. This configured must be done once per workspace (unless you use a unique workspace, or copy workspace configuration from one workspace to the next), but you don’t have to do it every time you start designer.

Configuring the processes

The next step is to import the designer projects (provided by Intalio) in Designer. To do so, in the top main menu, select *File -> Import*. In the window that comes up, select *General -> Existing Projects into Workspace* and click the *Next* button. Then, select the *Select Archive File* radio button and click on the *Browse* button. Now you need to select the archive containing the Designer projects provided by Intalio.

WARNING: Make sure you don’t confuse the Designer archive and the Server archive. Mixing them up will NOT work, as they contain very different files.

Once selected, you should see two projects in the pane below, SITA and TaskManager. Make sure both are selected and click on the *Finish* button. Both projects should now appear in the Project Explorer pane, in the top left section of Designer.

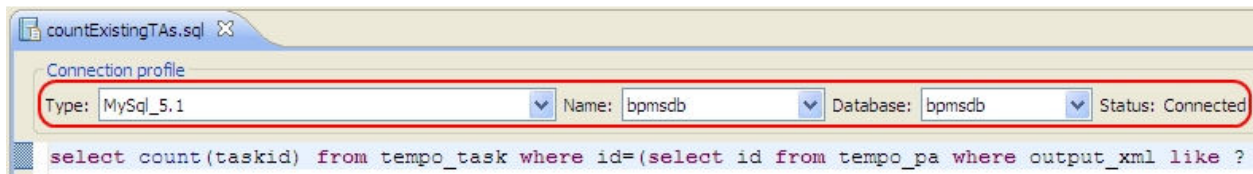
You can now work on the processes in Intalio Designer. Attending an Intalio public Enterprise Edition training is very useful, as this guide only covers the essentials of what you need to tweak this specific project, it doesn't go in-depth about the use of Intalio Designer.

WARNING: In each of your projects in Designer, you will have a directory called *build*. Never do anything in that directory unless you know exactly what you are doing, otherwise it could lead to problems. All the instructions you will find hereafter in this guide do not concern any file located in the *build* directory of any project.

Configuring the database connectors

Note: The configuration detailed in this section can be done at a later time, it will be detailed in the section called *Configuring the archived deployable projects*

One of the things you need to do is point the database connectors of in the SITA project to the corresponding databases. All the database connectors used in the SITA project are located in the *database* directory of the SITA project. Each and every one of them needs to be configured. To configure a database connector, open it in Intalio Designer, and at the top of the window you will have three fields: Type, Name, and Database. In Type, select *MySQL_5.1*. In Name, select the name that you gave to the corresponding connection. In database, select the name of the database. Once you have selected these three items, the status should read *Connected*. Please see the screenshot below for an example, the important part has been circled in red:



Don't forget to save each one of the connectors after configuring it. In the Process Explorer pane, there shouldn't be any warning signs (yellow triangle) on the database connector icons. If there still are some, go to the *Problems* view to see what the problem is.

There are different database connectors in the SITA project, which need to be associated to different databases:

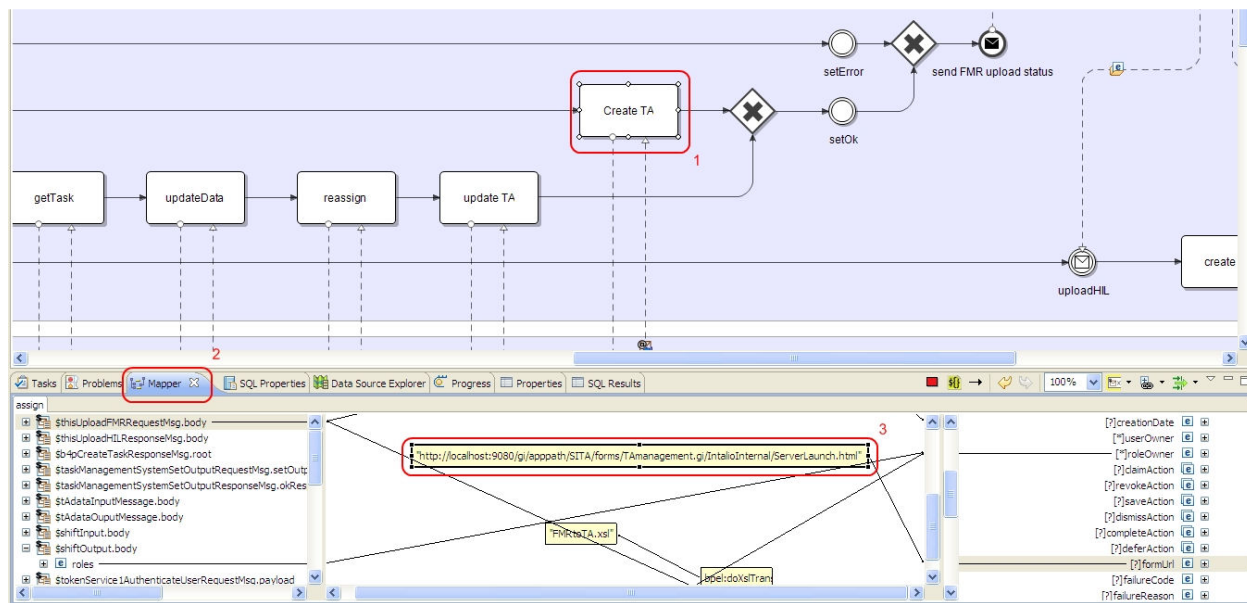
- countExistingTAs.sql needs to be associated to BPMSDB
- all connectors related to mechanics, avionics and coordinators (for example, getAllmechs.sql, getAllAvis.sql...), need to be associated to APPDATA
- storeFMR.sql needs to be associated to FMRLOG

Configuring the form endpoint

Note: The configuration detailed in this section can be done at a later time, it will be detailed in the section called *Configuring the archived deployable projects*

As mentioned before, in this project tasks don't belong to any project specifically, and because if that they are created "manually" by one of the processes, and this means that in this process the endpoint for the form is hardcoded. By default, that endpoint will be set to localhost, therefore if you are running designer and server on the same machine and using port 9080, you do not need to change anything. Nevertheless, if you are deploying to a remote server and/or on a different port, you will need to make this change.

In Designer, open the SITA project, and inside it open the directory called *process*. Inside this directory, open the diagram called *Upload.bpm*. In the diagram, there is a task called *Create TA*. Select it and open the mapper view. In the middle section of the mapper, you should see the endpoint, which should look like "[http://localhost:9080/qi/\[...\]/ServerLaunch.html](http://localhost:9080/qi/[...]/ServerLaunch.html)". You can edit it by double-clicking on the yellow box and changing the contents. You need to change the host name and port if you are not using port 9080 on your local machine. When you are done, you must save your diagram. Please find an explanatory screenshot below:



Configuring the WSDLs


Note: The configuration detailed in this section can be done at a later time, it will be detailed in the section called *Configuring the archived deployable projects*

You also need to configure web-service endpoints in the different projects. In the SITA project, open the *services* directory. In it, you will find several different WSDL files. Here is what you need to do with them:

- HHTtoRMA.wsdl, RMAnotif.wsdl, and TAlistToAMI.wsdl: these three WSDLs are the ones relative to the three JMS queues that RMA interacts with. If you are using activeMQ with the default configuration on the same machine as the one where Intalio server is running, then you must not make any modification in these WSDLs. Otherwise, you need to adapt the endpoint (host name and port) to match your activeMQ configuration
- All the OTHER WSDLs in the *services* directory: you need to set the endpoint for all the other WSDL files present in the *services* directory to match the configuration of the machine on which the Intalio server is running. That means that you must set the host name and port for each of these WSDL files to match that of the Intalio server that you are deploying to. Please note that one of the WSDLs, namely *TaskManagerProcess.wsdl*, has three different endpoints that must be all edited to the correct configuration.

There are also some WSDL endpoints to configure in the *TaskManager* project. Namely, there are three (once again, do not do anything in the *build* directory of any project), the endpoints must be set to the same host name and port as above. Please note that once again, *TaskManagerProcess.wsdl* has three different endpoints that must all be configured. Please also note that when you do so, designer will mention some errors in the *TaskManager* project. These errors are normal and will disappear once you have configured the deployment, which is covered in the next section of this guide.

Configuring the deployment

The last thing to configure is the deployment for each project. You can do so by clicking on the little gear button in the main Designer toolbar at the top: . The main thing you want to configure is the *Deployment URL*. You need to set that to the URL of the machine where Intalio server is running, for example <http://localhost:9080/ode> if you are running Intalio server on the same machine as Designer and on port 9080. You can also choose whether you want to deploy your project directly from designer to the server or archive it, in other words create an archive that contains the project ready to be deployed on the server by unzipping it in `<intalio>/var/deploy`. Please note that this is different from the export option that you can choose by selecting *File -> Export* from the main Designer toolbar, which exports the complete designer project, for example if you want to import it in another installation of Intalio Designer, on another machine for instance. Whichever option you choose (deploy, archive, or both), one you have selected it, you must save your deployment configuration. If you choose to archive your project, you can also set the *Export Directory*, which is where the archive will be stored after it is built. Before deploying/archiving, it is recommended to clean and build your project. To do so, in the process explorer pane (top-left section in Designer), right-click on your project directory and select the option *Clean and build now*. You need to do so for each project, after having configured and saved the deployment.

Once you have done so, you can click the *deploy* or *archive* button (depending which one you chose in the configuration) for each project. If you chose to deploy, the projects will be deployed to the server automatically, you should see a message saying *the deployment is successful* after several seconds. If you get an error message instead, that means there is an error somewhere in your project. Nevertheless, for the RMA, the archive option is better suited, as there are some editions to be made

after deployment. The following section describes what to do with the archives obtained when selecting the archive option for each project.

Configuring the archived deployable projects

Retrieve the TaskManager.zip and SITA.zip archive that have been obtained by deploying from Designer using the archive option. Unzip each of them to **an empty directory with the same name as the archive, without the .zip extension**. To perform the editions mentioned in this section, all you will need is a text editor, such as Notepad for instance. Nevertheless, a text editor with syntax coloration for XML such as Notepad++ or even an XML editor such as XMLspy is recommended as it will reduce the possibility of errors.

In the TaskManager process, there is only one thing to do: open the *deploy.xml* file which is in the *processes.ode* directory. In it, line 3 (right after the `<dd:deploy [...]><dd:process [...]>` statement), insert the following statement:

```
<dd:process-events generate="none"/>
```

Save the file. This statement disables the generation of BPEL events for this process. This means that BPEL events will not be generated and stored at runtime, therefore reducing resource consumption at the memory and database level. On the other hand, this reduces monitoring and troubleshooting possibilities, therefore if you are deploying to a test environment to do some testing, you may want to leave the file in its original state. Apply this modification when deploying to a production environment, when the projects have been fully tested.

In the SITA project, there are several things to do. The first one is analog to what you have done in the TaskManager project. Open the *deploy.xml* file in the *processes.ode* directory and insert the above statement after **each** of the `<dd:deploy [...]><dd:process [...]>` statements. This statement occurs as many times as there are different processes in the SITA project, therefore you must insert the above statement that same number of times at the corresponding places in *deploy.xml*. In the current state of the project, there are ten different processes in the SITA project, therefore you need to insert that statement ten times in the corresponding places.

In the SITA project, open the *forms.gi/forms* directory. Inside it, there are several directories named `<formName>.gi` where `<formName>` is the name of one of the forms in the project. In the latest state of the project, there are six different forms in the SITA project (there used to be more, some of which were never used and were therefore removed). In each of the `<formName>.gi` directory, there is a *rules* directory. Inside it, there can be XML files. Inside each of the XML files contained in each of the `<formName>.gi/rules` directories, you need to change every occurrence of "localhost:9080" without the quotes to whatever your server configuration is. For example, for the Lisbon environment, you need to change it to "172.17.128.120:9080" without the quotes. There should be two occurrences of this string in each XML file. In the current state of the project, there are four XML files in *TAmangement.gi/rules*, three XML files in *ShiftCreationLight.gi/rules* (these three are very recent, if there are none in your installation, don't worry about it), and one XML file in *uploadFMR.gi/rules*.

In the SITA project, open the *forms.gi/forms/TAmangement.gi* directory. Inside it, there are two things to do: In the designer project, in the *forms* directory, you may have noticed that there is a directory called *custom*. Inside this directory, there are two files: *appcanvas.xml* and *Packages.js*. You will be needing these two files in the next step:

- Open the *forms.gi/forms/TAmangement.gi/components* directory, and replace the existing **appcanvas.xml** with the one from the *custom* directory in the designer project
- Open the *forms.gi/forms/TAmangement.gi/IntalioInternal* directory and replace the existing **Packages.js** with the one from the *custom* directory in the designer project

The reason for doing this is that the behavior of the TAmangement.gi form is a little different from the default form behavior in Intalio, therefore these two files needed some customization. That is all for the *forms.gi* project, now return to the *SITA* directory, root of the unzipped server archive.

Note: the three following configurations may have been done previously in Designer. If that is the case, you don't need to do them again, but it is always a good idea to make sure before deploying.

- You may want to configure the database connectors, for instance if your Designer environment didn't have access to your server database and therefore couldn't be configured properly within Designer. To do so, open the *connectors.database/database* directory, you will notice that it contains as many directories as the number of database connectors in the SITA project, one per connector. Inside each of these directories there is a file called <connectorName>.sql.xml. In this file, you can configure the database connection parameters that the corresponding connector uses, for instance database URI, database Name, user name and password. As a reminder, please note that:
 - countExistingTAs.sql needs to be associated to BPMSDB
 - all connectors related to mechanics, avionics and coordinators (for example, getAllmechs.sql, getAllAvis.sql...), need to be associated to APPDATA
 - storeFMR.sql needs to be associated to FMRLOG
- Your WSDLs need to be configured with proper endpoints. From the SITA project root directory, go to the *processes.ode/services* directory. Inside it, there are several WSDLs, which all need to be edited. As a reminder:
 - HHTtoRMA.wsdl, RMAnotif.wsdl, and TAlistToAMI.wsdl: these three WSDLs are the ones relative to the three JMS queues that RMA interacts with. If you are using activeMQ with the default configuration on the same machine as the one where Intalio server is running, then you must not make any modification in these WSDLs. Otherwise, you need to adapt the endpoint (host name and port) to match your activeMQ configuration
 - All the OTHER WSDLs in the services directory: you need to set the endpoint for all the other WSDL files present in the *services* directory to match the configuration of the machine on which the Intalio server is running. That means that you must set the host name and port for each of these WSDL files to match that of the Intalio server that you are deploying to. Please note that one of the WSDLs, namely TaskManagerProcess.wsdl, has three different endpoints that must be all edited to the correct configuration.

- The form endpoint hardcoded in the Upload process must be changed with the proper server configuration. From the SITA project root directory, go to *processes.ode/process* and open Upload-UploadProcess.bpel. Inside it, search for “localhost:9080”, there should be one occurrence (careful, if you changed the endpoint in designer, it will be whatever you set it to instead). Change it to the correct server host name and port, keep the rest as it is.

Your processes are now fully configured and ready to be deployed on the Intalio server.

Deploying the processes

You now have two fully configured projects ready to be deployed on the Intalio server, SITA and TaskManager. Simply copy both of these directories and paste them in **<intalio>/var/deploy**. This has the same effect as deploying directly from Designer, except we have done some additional configurations in the previous steps that Designer wouldn't have done automatically.

Running the RMA

The RMA is now ready to be started. To start it, go to **<intalio>/bin** and run either the startup.bat or startup.sh script, depending on your operating system. After a minute or two, go to **<intalio>/var/deploy**, you should see two new files that have been generated, *SITA.deployed* and *TaskManager.deployed*. These two files mean that both projects are successfully deployed and ready to be run. If you have a file with the .invalid extension, that means there is a problem with the corresponding project. In that case, you can open the *.invalid file with a text editor and it will give you some information regarding the problem.

Note: if you make some changes to one of the projects' files, you need to re-deploy the process in order for these changes to be applied. To do so, simply delete the *.deployed file. It should be regenerated within a minute or so, meaning that the project has been redeployed.

Once the two projects are deployed correctly, you can access the user UI via any web browser at the following URL: <http://<host>:<port>/ui-fw> using the following credentials: login=admin, password=changeit. You can also monitor the different processes by going to the following URL: <http://<host>:<port>/bpms-console> using the same credentials.

The last thing you need to do is start the cyclic TA list export process: every three minutes, the complete list of active turnarounds is sent to AMI. It is an Intalio process that takes care of this, and it needs to be started once and for all (note that if you re-deploy the SITA project, you will need to start it again). To do so, go to the following URL with any web browser: <http://<host>:<port>/ui-fw> and log in with the credentials above. Open the *Processes* tab and click on *Start cyclic TA list export*. In the form that comes up, simply click on the *Start Process* button. After a few seconds, the screen should return to the task list. You can check that the process has been successfully started by logging in to

<http://<host>:<port>/bpms-console> and noting that there is one running instance (status is *in progress*) of that process.

Troubleshooting

There are different methods for troubleshooting issues in the RMA. The easiest one is to log in to bpms-console via a web browser and monitoring the processes. You can click on any instance and view the data it contains, view problems, and so on. Nevertheless, if BPEL events are deactivated, the use of this functionality will be reduced. Also, the cron-based instance cleanup will delete all *completed* instances, therefore you won't be able to monitor their data if this cleanup is activated.

For server issues, they will be logged in `<intalio>/var/log/bpms.log`. You can configure the categories that are logged by editing `<intalio>/var/log/log4j.properties`, nevertheless please note that any changes you make in this file will only be applied after a server restart.

You can switch FMR logging ON or OFF by changing the value of the `fmrlog` property in `<intalio>/conf/sita.properties`. Please see an earlier section of this guide for more precisions.

For any additional question you can contact Intalio support using the SITA support account at the following URL: <http://support.intalio.com>, and for questions relating to this project specifically, please contact Pierre Pavageau: pavageau@intalio.com.

Appendix: cron-based instance cleanup

This section provides detailed information on how to configure the cron-based instance cleanup in Intalio server. Please note that any information written in any previous section of this guide takes precedence over any information written in this appendix.

System cron settings take the configuration from the `schedules.xml` file under the ODE configuration directory. The ODE configuration directory is configured through a JVM system property: `org.apache.ode.configDir`.

e.g.

```
-Dorg.apache.ode.configDir=$CATALINA_HOME/var/config
```

In the following example `schedules.xml` file, runtime data for any process is cleaned up every midnight if the associated process instance

1. was last active more than one day ago and has been completed, or,

2. was last active more than one week ago and has been faulted or failed.

e.g.

```
<?xml version='1.0' encoding='UTF-8'?>

<sched:schedules
  xmlns:sched="http://www.apache.org/ode/schemas/schedules/2009/05"
  xmlns:dd="http://www.apache.org/ode/schemas/dd/2007/03">
  <sched:schedule when="0 0 0 * * ?">
    <sched:cleanup>
      <dd:filter><![CDATA[last_active<-1d
status=completed]]></dd:filter>
      <dd:filter><![CDATA[last_active<-1w
status=active|suspended|completed|terminated|failed]]></dd:filter>
    </sched:cleanup>
  </sched:schedule>
</sched:schedules>
```

- Use one or more spaces between filters when combining them with the 'and' logic. For 'or' logic, list multiple <filter> elements.
- Use multiple <schedule> elements to clean up runtime data using different filters at different times.
- Use the cleanup categories to clean up only the specified types of data.

In the following example, all events are cleaned up at least 1 hour later after the process instance completes. The rest of runtime data is cleaned up at least 1 day later.

e.g.

```
<?xml version='1.0' encoding='UTF-8'?>

<sched:schedules
  xmlns:sched="http://www.apache.org/ode/schemas/schedules/2009/05"
  xmlns:dd="http://www.apache.org/ode/schemas/dd/2007/03">
  <sched:schedule when="0 0 * * * ?">
    <sched:cleanup>
      <dd:filter><![CDATA[last_active<-1h
status=completed]]></dd:filter>
      <dd:category>events</dd:category>
    </sched:cleanup>
  </sched:schedule>
  <sched:schedule when="0 0 0 * * ?">
    <sched:cleanup>
      <dd:filter><![CDATA[last_active<-1d
status=completed]]></dd:filter>
    </sched:cleanup>
  </sched:schedule>
</sched:schedules>
```

Process cron configuration

Each process can have its own cron settings for runtime data cleanup in its deployment descriptor file. A process that has its own cleanup cron settings is excluded from the system cron cleanup operations.

In the following example deploy.xml file, runtime data for the process is cleaned up if the associated process instance was last active more than one month ago and has been completed or terminated.

```
<dd:schedule when="* * * * * ?">
  <dd:cleanup>
    <dd:filter><![CDATA[last_active<-1m
status=completed|terminated]]></dd:filter>
  </dd:cleanup>
</dd:schedule>
```

You can combine cron-based clean up configurations with the on-the-fly instance clean up. In the following example, messages, correlations and events are deleted as soon as a process instance is completed, faulted or terminated. If the instance was completed or terminated, any remaining runtime data for the instance is cleaned up one day later.

```
<deploy xmlns="http://www.apache.org/ode/schemas/dd/2007/03"
  xmlns:pns="http://ode/bpel/unit-test"
  xmlns:wns="http://ode/bpel/unit-test.wsdl"
  xmlns:dns="http://axis2.ode.apache.org">

  <process name="pns:HelloWorld2">
    <active>true</active>
    <provide partnerLink="helloPartnerLink">
      <service name="wns:HelloService" port="HelloPort"/>
    </provide>
    <invoke partnerLink="dummyPartnerLink">
      <service name="dns:DummyService"
port="DummyServiceSOAP11port_http"/>
    </invoke>
    <cleanup on="always">
      <category>messages</category>
      <category>correlations</category>
      <category>events</category>
    </cleanup>
    <dd:schedule when="* * * * * ?">
      <dd:cleanup>
        <dd:filter><![CDATA[last_active<-1d
status=completed|terminated]]></dd:filter>
      </dd:cleanup>
    </dd:schedule>
  </process>
</deploy>
```

Cron expression

Fields in ODE cron expression start from the second value.

```
.----- second (0 - 59)
| .----- minute (0 - 59)
| | .----- hour (0 - 23)
| | | .----- day of month (1 - 31)
| | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
| | | | | .---- day of week (0 - 6) (Sunday=0 or 7) OR
sun,mon,tue,wed,thu,fri,sat
| | | | |
* * * * *
```

e.g. every day, at 1 min past the midnight

```
0 1 0 * * ?
```

e.g. every two hours at the 13th minute

```
0 13 */2 * * *
```

Cleanup configuration filters

- **pid** : the process id of a process
e.g. collect all runtime data associated with a process id

```
pid={http://example.com/HelloWorld/HelloWorld}HelloWorld-2
```

- **namespace** : the namespace of a process
e.g. collect all runtime data associated with a process using the name space

```
http://example.com/HelloWorld/HelloWorld
```

e.g. collect all runtime data associated with a process using the name space and with a wild card

```
http://example.com/HelloWorld/*
```

- **name** : the name of the process
e.g. collect all runtime data associated with a process using the local part

```
HelloWorld-2
```

e.g. collect all runtime data associated with a process using the local part and with a wild card

```
HelloWorld*
```

- **status** : the status of a process instance
Valid status are active, suspended, completed, terminated or failed. You can or multiple status

with the '|' symbol.

e.g. if the status of a process instance is completed or terminated

```
completed|terminated
```

- **started** : the datetime when a process instance started
e.g. filter runtime data associated with a process instance that started at least 1 year ago

```
started<=-1y
```

e.g. filter runtime data associated with a process instance that started more than 3 months ago

```
started<-3M
```

e.g. filter runtime data associated with a process instance that started more than 1 week ago

```
started<-1w
```

e.g. filter runtime data associated with a process instance that started more than 1 day ago

```
started<-1d
```

e.g. filter runtime data associated with a process instance that started more than 12 hours ago

```
started<-12h
```

e.g. filter runtime data associated with a process instance that started more than 30 mins ago

```
started<-30m
```

e.g. filter runtime data associated with a process instance that started more than 60 seconds ago

```
started<-60s
```

e.g. filter runtime data associated with a process instance that started before year 1997

```
started<1997
```

e.g. filter runtime data associated with a process instance that started before July, 1997

```
started<1997-07
```

e.g. filter runtime data associated with a process instance that started before July 16, 1997

```
started<1997-07-16
```

e.g. filter runtime data associated with a process instance that started before the date and time

```
started<1997-07-16T19:20+01:00
```

e.g. filter runtime data associated with a process instance that started before the date and time

```
started<1997-07-16T19:20:30+01:00
```

- **last_active** : the last active datetime of the process instance
The **last_active** field supports all the date formats that the 'started' field supports.
e.g. filter runtime data associated with a process instance that was last active more than 1 week ago

```
started<-1w
```

Note: Setting the **last_active** filter to a reasonable value is highly recommended; depending on what dbms you're using, deleting rows in a different thread while one thread holds the db objects may generate excessive locks and/or deadlocks.

Troubleshooting

Lock time-outs and/or deadlocks in the dbms

Add the **last_active** filter. Try setting it to a reasonably big value such as -1m or -1d.

JTA Transaction time-outs

The cron-based instance data cleanup is a database intensive batch operation. The default transaction time-out value for the Intalio Server might not be long enough. Try setting it to a bigger value. For instance, with the Bitronix transaction manager, add the following line in the `btm-config.properties` file:

```
bitronix.tm.timer.defaultTransactionTimeout=300
```

Cleaning up cannot keep up with generation of data

1. Process instances are deleted by batch and the default batch size is 10 instances. You can increase this value up to a few hundreds through a JVM system property, in the `<bpms>/bin/setenv.sh` file (on Windows, `<bpms>/bin/setenv.bat`).

```
-Dorg.apache.ode.processInstanceDeletion.transactionSize=300
```

2. Depending on the design of your process, there could be too many events generated. Disable the event generation:

```
<dd:process name="HelloWorld">  
  <dd:process-events generate="none"/>
```

You need to add the XML element to each process in the `deploy.xml`.