
HighQSoft



ASCOBA

Version 1.1

ASam COpy and BAckup

[Karst Schaap](#) 14.08.2008

Contents

1	Introduction	1
1.1	Introduction to ASCOBA second generation	1
2	Model Mapper tutorial	3
2.1	Model mapper tutorial.	3
2.2	The properties in the job file.	3
2.3	Mail	4
2.4	Object definition sequence	4
2.5	The tags of the output devices.	4
2.6	The initialization	6
2.7	Standard output	7
2.8	Standard Error	7
2.9	Engine	7
2.10	The rules	7
2.11	Conclusion	9
2.12	The terminate	9
3	The implementations	11
3.1	Introduction to ASCOBA implementations	11
3.2	CompareModel	11
3.3	CopyModel	11
3.4	OdsCache	13
4	The rules	15
4.1	Introduction to ASCOBA rules	15
4.2	AddLiteralRule	15
4.3	AscobaAbstractRule	15
4.4	AscobaInputRule	16
4.5	CopyInstancesAbstractRule	16

4.6	CopyInstancesChildrenRule	18
4.7	CopyInstancesRule	18
4.8	CopyInstancesWithFatherChildRule	18
4.9	CopyInstancesWithRelatedRule	19
4.10	CopyInstancesWithRelationsRule	19
4.11	CSVFromRule	20
4.12	CSVToRule	21
4.13	DeleteInstancesRule	22
4.14	DistinctRule	22
4.15	ElementToRule	22
4.16	HandledInstancesRule	24
4.17	HQLXRule	24
4.18	LinkInstancesRule	25
4.19	RelatedInstancesRule	25
4.20	RelationToRule	26
4.21	RenameRule	26
4.22	SetSystemPropertyRule	26
4.23	SetUndefinedRule	27
4.24	StoreRule	27
4.25	WriteAttributeRule	28
4.26	XLSDate2ODSRule	28
4.27	XLSFromRule \addindex XLS \addindex from rule XLS	28
5	The TestModel	31
5.1	The application model of the test.	31
6	Example files	33
6.1	The Athos INI-File.	33
6.2	An ASCOBA example file.	34
7	Appendix	41
7.1	Regular expression	41
8	Modification history	43

Chapter 1

Introduction

1.1 Introduction to ASCOBA second generation

ASCOBA, (ASam COpy and BAckup) the first generation have a fixed rules to follow the relations, this work well in a lot of cases but is fixed and do not work in all cases. ASCOBA the second generation is based on the model mapper, based on the schema, and have there for no rules at all. There are some rules implemented which can be used to configure the implementation of ASCOBA but it is not required to use these rules and it is easy to extend the rules by implementing a new rule for customizer special problems.

In **Model mapper tutorial**.(p. 3) is a description how to use the Model Mapper. The XML-schema for the of the Model Mapper the current documentation is given in the at [Schema Definitions](#)

ASCOBA can be started with the following command `JRE_HOME%\bin\java -jar ascoba2g.jar ascoba.xml file.atfx` The java version must be Java 1.5.



Chapter 2

Model Mapper tutorial

2.1 Model mapper tutorial.

Setting up an importer with the XML-based ModelMapper.

First create a job XML-File, the job file is defined with the tag `<job>`. The `schemaLocation` allows validating of the XML-file and is helpful for editing.

```
<job xmlns="http://www.highqsoft.de/schema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.highqsoft.de/schema http://www.highqsoft.serveftp.net/schema/schema.xsd"
      gathermode="PEDANTIC"
      messagelevel="ALL"
      tracelevel="ALL"
      loglevel="ALL">
</job>
```

for the attributes of the job see the schema documentation.

2.2 The properties in the job file.

The environment can be used as properties within the job file.

```
<!-- The property definitions. -->
<environment name="env"/>
```

Any environment variable can be used with the prefix 'env.' as a property in the job file.

The properties which are defined in the job file must be include in the tag `<properties>` each property is defined in an own tag `<property>`. A set of properties can be defined in an external file, and the job-file defined only the prefix of these properties.

```
<properties>
  <!--
    The common properties.
  -->
  <property name="Setup." file="${basedir}/setup.properties"/>
```

```

<!--
    ODS Server
-->
<property name="ODSServerService" value="${Setup.ServerService}"/>
<property name="ODSServerHost" value="${Setup.ServerName}"/>
<property name="ODSServerPort" value="${Setup.ServerPort}"/>
<property name="ODSServerUser" value="${Setup.ServerUser}"/>
<property name="ODSServerPassword" value="${Setup.ServerPassword}"/>

<!--
    XATF Reader
-->
<property name="XATFSourceRef" value="PAC_ATF_XML"/>

</properties>

```

Note:

in the file `setup.properties` in current directory must have the properties `ServerService`, `ServerName`, `ServerPort`, `ServerUser`, `ServerPassword`. The user is responsible for these information in the file. When the property does not exist the software can throw an exception or write a message.

The use of the properties in the code is `String sourceRef = ctx.getProperty(XATFSourceRef);`, the properties are automatically substituted. The context `ctx` is set by the Moma in the implementation with `setContext` from the interface `com.highqsoft.xsd.ImplementationIF`

2.3 Mail

You can define a mail address, where send the mail during the execution.

2.4 Object definition sequence

The object definition sequence, I don't know how to use.

2.5 The tags of the output devices.

In a job file tree kind of output devices can be defined, the logger, tracer and messenger. Only when an output device is defined the messages to the output defined will be reported. The attributes at the job defines which kind of messages are reported. The order of the tags is `<logger>`, `<tracer>` followed by `<messenger>`.

2.5.1 The `<logger>` tag

The logger reports the logging messages. These are normally the messages for the logging, such which file is handled. The state of the last operation.

All information for customer is required.

```

<logger name="com.highqsoft.xsd.DefaultHandler">
  <formatter name="com.highqsoft.xsd.DefaultFormatter">
    <arguments>

```



```

    <property name="FORMAT" value="{0,date} {0,time}. {1,number,###} diff: {2,number,####} Thread 0x{3} {6} {4}
    <property name="LINEBREAK" value="&#13;&#10;&#9;" />
    <property name="SEPARATOR" value="&#13;&#10;*****" />
  </arguments>
</formatter>
</logger>

```

The class `DefaultHandler` prints the output to the standard error on the console. There is also a file based hadler which will store the messages in a file. The class is `DefaultFileHandler`. The followin example shows how this ane is configured.

```

<!-- The logger -->
<logger name="com.highqsoft.xsd.DefaultFileHandler">
  <formatter name="com.highqsoft.xsd.DefaultFormatter">
    <arguments>
      <property name="FORMAT" value="{0,date} {0,time}. {1,number,###} diff: {2,number,####} Thread 0x{3} {6} {4}
      <property name="LINEBREAK" value="&#13;&#10;&#9;" />
      <property name="SEPARATOR" value="&#13;&#10;*****" />
    </arguments>
  </formatter>
  <arguments>
    <property name="FILENAME" value="\${basedir}/logging.log"/>
  </arguments>
</logger>

```

A special logger it must extends the abstract class `java.util.handler.Handler` and implements the interface `com.highqsoft.xsd.ImplementationIF` the classname can be given at the 'name' attribute. The formatter is used to format the messages. A special formatter must extend the abstract class `java.util.logging.formatter`.

```

// Get the logger
Logger logger = ctx.getLogger();
// Check the log level
if (logger.isLoggable(LogLevel.STATUS)) {
  String message = ctx.getStatusMessage();
  if (message == null) message = "";
  // Format the message
  String logMsg = MessageFormat.format(ctx.getProperty("Transformer.TearDown"),
    new Object[]{ctx.getStatus().name(), time, message});
  // Log the message
  logger.log(LogLevel.STATUS, logMsg);
}

```

2.5.2 The <tracer> tag

The tracer reports the tracing messages. These are normally the messages for the trace to the called methods. The tracer is identical with the logger. Each method where the proces enter and leave, not very useful because the usage of development tools.

2.5.3 The <messenger> tag

The messenger reports the messages. These are normally the messenger all occuring messages. The messenger is identical with the logger and tracer. The information for the developer, to see the how the proces run, which data is used.

2.6 The initialization

The initialisation has 5 optional tags in a fix order the tags are <startup>, <source>, <loading>, <target> and <terminate>. The source and target tag are always for an ASAM ODS session.

2.6.1 The <startup> tag.

The startup tag is to initialize the used implementations. The startup is implementation is called before the source is opened. The methods `init()`, `execute()` and `terminate()` are called in exactly this order. Any implementation must implements the interface `com.highqsoft.xsd.ExecutableIF`.

```
<init>
  <startup>
    <implementation name="com.highqsoft.ascoba.CompareModel">
      <arguments>
        .....
      </arguments>
    </implementation>
  </startup>
</init>
```

This implementation have two arguments, these arguments are implementation dependent and must be documented at the implementation. with the following code the `String filename = ctx.removeDecorator(arguments.getProperty("XATFFILENAME"))`; In case the implementation is derived from the class `com.highqsoft.xsd.AbstractImplementation` all argument names are uppercase. the method `arguments.getProperty("XATFFILENAME")` returns the aragument as given in the job file, the method `ctx.removeDecorator()` substitute the properties given in the property value by the real value.

The block with the implementation can be repeated.

To store the implmentation object in the context the following code is used `ctx.saveReference(refName, this)`; The `refName` is the name used to get the context. Other implmentations (such as rules) can get the implmentation object by using the method `restoreReference()`

2.6.2 The <source> tag

With the source tag only an ASAM ODS session can be defined.

```
<source>
  <aosession user="{ODSServerUser}" password="{ODSServerPassword}" id="source">
    <aofactory servicename="{ODSServerService}">
      <aoservice plugablename="CORBA" nameservicehost="{ODSServerHost}"
        nameserviceport="{ODSServerPort}"/>
    </aofactory>
  </aosession>
</source>
```

With the following code `org.asam.ods.AoSession aoSess = (org.asam.ods.AoSession) ctx.restoreReference("source")`; can the session be loaded in other implementations. The key must correspond with the name given at the attribute 'id'.

The ODS Session is automatic closed at the end of the job, if there is any transaction open (headtransaction) in the ODS session and the session close, depending on the INI-File variable `SESSION_CLOSE_COMMIT` the session is committed or aborted.

2.6.3 The <loading> tag

The <loading> tag is identical with the <startup> tag, the implementation are only called after the source is opened.

2.6.4 The <target> tag

The <target> tag is identical with the <source> tag, the ASAM ODS session is stored in the context with the name 'target'.

2.6.5 The <terminate> tag

The <terminate> tag is identical with the <startup> or <loading> tag, the implementation are only called after the target is opened.

2.7 Standard output

Allowed to redirect the standard output device.

2.8 Standard Error

Allowed to redirect the standard error device.

2.9 Engine

Allow to overwrite the implementation of the rule engine.

2.10 The rules

The rules are compined in a tag <rulesdef>. At the definition of the rules the decision can be made which types of rules are the driving rules. There are fromrules tag <fromrules> and torules tag <torules>, each one is a sequence of rules tag <ruleseq>. Each rule has a 'id' and the 'pointTo' rule can be defined. The rule engine will always call the method `getValue()` at the fromrules and the ouput of this method is used as parameter at the method `setValue()` at the torules. When there is no 'pointTo' rule defined only the rule is executed and no second rule is executed. The 'Id' of the rule must not be unique. In case from the driving rule different rules are driven all driven rules must have the same Id.

```
<rulesdef drivenby="FROM">
  <fromrules>
    <ruleseq>
      <rule id="from" pointto="to">
        <procedure name="com.highqsoft.schema.FromRule">
          </procedure>
        </rule>
      </rule>
    </ruleseq>
  </fromrules>
```

```

<torules>
  <ruleseq>
    <rule id="to">
      <procedure name="com.highqsoft.schema.ToRuleA">
        </procedure>
      </rule>
    <rule id="to">
      <procedure name="com.highqsoft.schema.ToRuleB">
        </procedure>
      </rule>
    <rule id="to">
      <procedure name="com.highqsoft.schema.ToRuleC">
        </procedure>
      </rule>
    </ruleseq>
  </torules>
</ruledef>

```

In the example above the fromrule 'from' points to the rules with the id 'to'. The rule engine will execute first the method `getValue()` of the class 'FromRule' then execute the method `setValue` of the classes 'ToRuleA', 'ToRuleB' and 'ToRuleC'.

The methods `getValue()` and `setValue()` have a `NameValueUnit[][]` as return or input parameter. The `NameValueUnit` is the attribute name, value with unit combination. The field `valName` is the attribute name, the field `value` is the value and the field `unit` is the unit of the value. The first sequence of `NameValueUnit` are the attributes of one instance. The second sequence are the sequence of instances, it is not guarantee that all instances have the same amount of attributes. The total is also called a dataset. The structure `NameValueUnit` is defined in the ASAM ODS OO-API.

2.10.1 The <rule> tag.

One rule has always an id, the id is the identifier of the rule. The rule is searched by the identifier.

There are several default rules defined such as 'literal', 'attribute' etc. The 'procedure' rule is a special rule because there is no implementation available. The class given with the 'name' must implements `com.highqsoft.xsd.RuleIF` and can extends `com.highqsoft.xsd.AbstractRule`.

```

<rule id="toDim" sessionref="target">
  <headtransaction type="START"/>
  <procedure name="com.highqsoft.schema.ToRule">
    <arguments>
      <property name="ReferenceName" value="{ODSTargetRef}"/>
      <property name="ElementName" value="dim"/>
    </arguments>
  </procedure>
</rule>

```

Further shows the example the tag `<headtransaction>` with the `type="START"`. This rule starts the transaction of the `sessionref` before the rule is executed (`setValue()`).

The procedure rules have a tag `<arguments>` these arguments are set to the rule with the method `setArguments()` and the arguments are available at the initialization (`init()`) of the rule.

Note:

The transaction is not aborted when an error occurs, the session is only closed and the server decides whether to commit or abort the session.

2.11 Conclusion

The conclusion is another section in the job-file, in this section there can be defined also implementation like at init.

An example is given below

```
<implementation name="com.highqsoft.schema.ReturnWriter">
  <arguments>
    .....
  </arguments>
</implementation>
```

2.12 The terminate

This implementation is called when the source and target are closed.



Chapter 3

The implementations

3.1 Introduction to ASCOBA implementations

The implementations are given in this section. The implementations used also the argumentes of the extended implementations.

3.2 CompareModel

This implementation compare the model of the ASAM ODS source with the model of the ASAM ODS target. All elements with all there attributes, all relations and all enumerations are compared.

The following table lists all the argument of this implementation. Keywords are case insensitive.

Argument Keyword	Datatype	Default	Description
sourcesession	String	<i>source</i>	The name of the ASAM ODS source.
targetsession	String	<i>target</i>	The name of the ASAM ODS target.

An example of the usage of the implementation is given below:

```
<terminate>
  <implementation name="com.highqsoft.ascoba.CompareModel">
    <arguments>
      <property name="sourcesession" value="source"/>
      <property name="targetsession" value="target"/>
    </arguments>
  </implementation>
</terminate>
```

3.3 CopyModel

This implemenmtation copies the model of the ASAM ODS source to an ASAM ODS target. All elements with all there attributes, all relations and all enumerations are copied. The instance of AoEnvironment will be copied also. The security is copied when the argument is given. All instances of AoUserGroup and the security information, security level the rights and initial rights and the initial right relations, of the application model is copied from the source to the target.

The copy of the model is done in an own transaction, external transaction control is not possible.

The Id of the copied instances from the stored will be remembered in the implementation.

The following table lists all the argument of this implementation. Keywords are case insensitive.

Argument Keyword	Datatype	Default	Description
sourcesession	String	<i>source</i>	The name of the ASAM ODS source.
targetsession	String	<i>target</i>	The name of the ASAM ODS target.
withsecurity	boolean	<i>false</i>	Copy the security information of the application model from the source to the target. The instances of the AoUserGroup are all copied.
copyInstance-Attributes	boolean	<i>false</i>	Copy the instance attributes when they exist.
useGetValueSeq	boolean	<i>true</i>	Use the method <code>getValueSeq</code> to load the values of all attributes of an instance with one call from the source session. There are some server implementations which have not implement this method. When this flag is set to false the method <code>getValue</code> is used for each attribute which will reduce the performance.
skipId	boolean	<i>false</i>	Skip the Id Attribute, when the target is a storage (Database) which creates new Id the Id must be skipped, otherwise when the original Id is required the Id (write to file) is required and should not be skipped.

An example of the usage of the implementation is given below, the source and target session are given, the implementation copies the application model together with the security from the source

to the target. The `source` is an CORBA-service, the target is an ATF/XML-service and the filename is given in the first argument of the calling program.

```
<init>
  <source>
    <aosession user="karst" password="secret" id="source">
      <aofactory servicename="TestModel/DB">
        <aoservice plugablename="CORBA" nameservicehost="TestServer" nameserviceport="2809"/>
      </aofactory>
    </aosession>
  </source>

  <target>
    <aosession filename="${ARGV[1]}" openmode="w" id="target">
      <aofactory servicename="XATF/File">
        <aoservice plugablename="ATHOS" inifile="${ATHOS_INI}"/>
      </aofactory>
    </aosession>
  </target>

  <terminate>
    <implementation name="com.highqsoft.ascoba.CopyModel">
      <arguments>
        <!-- The properties sourcesession and targetsession uses the default values -->
        <property name="WithSecurity" value="true"/>
      </arguments>
    </implementation>
  </terminate>
</init>
```

3.4 OdsCache

This is the cache object for the ODS Session. The application structure is cached, with the application elements there base elements, there application attributes and the relation between the elements.

The Id of the instance element are cached. An identifier (Element + Name) is the key for the Id of the instance element in the cache.

The following table lists all the argument of this implementation. Keywords are case insensitive.

Argument Keyword	Datatype	Default	Description
ReferenceName	String	<i>none</i>	The name of reference where the object must be stored in the context.
OdsSession	String	<i>none</i>	The reference name of the ODS session.

An example of the usage of the implementation is given below:

```
<terminate>
  <!-- The initialization of the ODS Cache, use the target as ODS session. -->
  <implementation name="com.highqsoft.ascoba.OdsCache">
    <arguments>
      <property name="ReferenceName" value="${ODSTargetRef}"/>
      <property name="OdsSession" value="target"/>
    </arguments>
  </implementation>
</terminate>
```



Chapter 4

The rules

4.1 Introduction to ASCOBA rules

The implemented rules are given in this section. The rules used also the argumentes of the extended rules.

4.2 AddLiteralRule

This rule will add to each row of the input of the `setValue` method a column with the given name and value.

This rule can be used as a sequence of rules.

The following table lists all the argument of this implementation. Keywords are case insensitive.

Argument Keyword	Datatype	Default	Description
AttributeName	String	<i>none</i>	The name of the attribute.
AttributeValue	String	<i>none</i>	The value of the attribute.
AttributeDatatype	DataType	<i>DT_STRING</i>	The datatype of the attribute value.

4.3 AscobaAbstractRule

This rule is an abstract rule for ASCOBA.

The following table lists all the argument of this implementation. Keywords are case insensitive.

Argument Keyword	Datatype	Default	Description
sourcesession	String	<i>source</i>	The name of the ASAM ODS source.
elementname	String	<i>none</i>	The name of the application element in the ASAM ODS service.
elementId	String	<i>none</i>	The Id of the application element in the ASAM ODS service.

4.4 AscobaInputRule

This rule will return a data set with the Id of instance given from the input. The application element of the source is used.

The following table lists all the argument of this implementation. Keywords are case insensitive.

Argument Keyword	Datatype	Default	Description
instanceId	String	<i>none</i>	The Id of the instance element in the ASAM ODS service.

Futher arguments are inherit from the super class.

An example of the usages of this rule is given below.

```
<rule id="getElement" pointto="copyElement">
  <procedure name="com.highqsoft.ascoba.AscobaInputRule">
    <arguments>
      <property name="ElementName" value="${SelectElement}"/>
      <property name="InstanceId" value="${InstanceMeaId}"/>
      <property name="sourcesession" value="source"/>
    </arguments>
  </procedure>
</rule>
```

4.5 CopyInstancesAbstractRule

This rule is an abstract rule to copy instances from the source to the target. The source and target application model must be more or less identical. The attributes of the target application element must be available in the source application model. This rule extends **AscobaAbstractRule**(p. 15)

The following table lists all the argument of this implementation. Keywords are case insensitive.



Argument Keyword	Datatype	Default	Description
targetsession	String	<i>target</i>	The name of the ASAM ODS target.
withsecurity	boolean	<i>false</i>	Copy the security information of the application model from the source to the target. The instances of the AoUserGroup are all copied.
copyInstance-Attributes	boolean	<i>false</i>	Copy the instance attributes when they exist.
useGetValueSeq	boolean	<i>true</i>	Use the method <code>getValueSeq</code> to load the values of all attributes of an instance with one call from the source session. There are some server implementations which have not implement this method. When this flag is set to <i>false</i> the method <code>getValue</code> is used for each attribute which will reduce the performance.
skipId	boolean	<i>false</i>	Skip the Id Attribute, when the target is a storage (Database) which creates new Id the Id must be skipped, otherwise when the original Id is required the Id (write to file) is required and should not be skipped.

Further arguments are inherit from the super class.

The application model used is documented at **The application model of the test.**(p. 31)

When the security is copied the rights and initial rights of the instances are copied. The rule expects the instances of the AoUserGroup are available in the target.

4.6 CopyInstancesChildrenRule

This rule copies the children of the instances from the source to the target session. The rule extends **CopyInstancesAbstractRule**(p. 16). This rule is a **TO-Rule**, so only the method `setValue()` is implemented.

The only children instances are copies also.

The security information can be copied depending of the arguments.

There is no transaction handling in this rule, the transaction must be started at the *TargetSession* and must be committed outside this rule.

The element Id's of the copied instances from the source are remembered in the implementation.

The arguments are inherit from the super classes.

4.7 CopyInstancesRule

This rule copies the instances from the source to the target session. The rule extends **CopyInstancesAbstractRule**(p. 16). This rule is a **TO-Rule**, so only the method `setValue()` is implemented.

The Id of the instances is given as input at the method `setValue()`. The element is given as the argument `elementName`. No related instances are copied.

The security information can be copied depending of the argument.

There is no transaction handling in this rule, the transaction must be started at the *TargetSession* and must be committed outside this rule.

An example of the usage of the rule is given below, the instances of the element `Log` are copied with security from the source to the target

```
<rule id="copyMea">
  <procedure name="com.highqsoft.ascoba.CopyInstancesRule">
    <arguments>
      <property name="ElementName" value="Log"/>
      <property name="sourcesession" value="source"/>
      <property name="targetsession" value="target"/>
      <property name="WithSecurity" value="true"/>
    </arguments>
  </procedure>
</rule>
```

The element Id's of the copied instances from the source are remembered in the implementation.

The arguments are inherit from the super classes.

4.8 CopyInstancesWithFatherChildRule

This rule copies the instances from the source to the target session. The rule extends **CopyInstancesAbstractRule**(p. 16). This rule is a **TO-Rule**, so only the method `setValue()` is implemented.

The parent(father) and children instances are copies also.

The security information can be copied depending of the arguments.

There is no transaction handling in this rule, the transaction must be started at the *TargetSession* and must be committed outside this rule.

The element Id's of the copied instances from the source are remembered in the implementation.

The arguments are inherit from the super classes.

4.9 CopyInstancesWithRelatedRule

This rule copies the instances from the source to the target session. The rule extends **CopyInstancesAbstractRule**(p. 16). This rule is a **TO-Rule**, so only the method `setValue()` is implemented.

The instance element is copied, the parent and child instances are copied also. The `INFO_TO` related instances are copied also, with the parent instance but not with their children.

The Id of the instances is given as input at the method `setValue()`. The element is given as the argument `elementName`.

The security information will be copied depending of the arguments.

There is no transaction handling in this rule, the transaction must be started at the *TargetSession* and must be committed outside this rule.

An example of the usage of the rule is given below, the instances of the element `Measurement` are copied with security from the `source` to the `target`

```
<rule id="copyMea">
  <procedure name="com.highqsoft.ascoba.CopyInstancesWithRelatedRule">
    <arguments>
      <property name="ElementName" value="Measurement"/>
      <property name="sourcesession" value="source"/>
      <property name="targetsession" value="target"/>
      <property name="WithSecurity" value="true"/>
    </arguments>
  </procedure>
</rule>
```

When this rule is used on the `TestModel` the instance of the element `Measurement`, the instances of the parent elements `SubTest` and `Test` are copied. The instances of the child elements `SubMatrix`, `MeasurementQuantity`, `LocalColumn` and `ExternalComponent` are copied. The instances of the elements related with the relationship `INFO_TO` are copied also, these are the elements `Vehicle` and `TestBed` to the element `Measurement`. The instances of the elements `Quantity` and `Unit` to the element `MeasurementQuantity` and the instances of the element `PhysicalDimension` related to the element `Unit`. The relations points to the elements which are copied. The instances of the child elements of the related elements are not copied, in the `TestModel` the instances of the elements `TestBedPart` or `Engine`.

The element Id's of the copied instances from the source are remembered in the implementation.

The arguments are inherit from the super classes.

4.10 CopyInstancesWithRelationsRule

This rule copies the instances from the source to the target session. The rule extends **CopyInstancesAbstractRule**(p. 16). This rule is a **TO-Rule**, so only the method `setValue()` is implemented.

The instance element is copied, the parent and child instances are copied also. The INFO_TO related instances are copied also, with the parent instance but not with their children.

The Id of the instances is given as input at the method `setValue()`. The element is given as the argument `elementName`.

The security information will be copied depending of the arguments.

There is no transaction handling in this rule, the transaction must be started at the *TargetSession* and must be committed outside this rule.

An example of the usage of the rule is given below, the instances of the element `Measurement` are copied with security from the `source` to the `target`

```
<rule id="copyMea">
  <procedure name="com.highqsoft.ascoba.CopyInstancesWithRelationsRule">
    <arguments>
      <property name="ElementName" value="Measurement"/>
      <property name="sourcesession" value="source"/>
      <property name="targetsession" value="target"/>
      <property name="WithSecurity" value="true"/>
    </arguments>
  </procedure>
</rule>
```

When this rule is used on the TestModel the instance of the element `Measurement`, the instances of the parent elements `SubTest` and `Test` are copied. The instances of the child elements `SubMatrix`, `MeasurementQuantity`, `LocalColumn` and `ExternalComponent` are copied. The instances of the elements related with the relationship INFO_TO are copied also, these are the elements `Vehicle` and `TestBed` to the element `Measurement`. The instances of the elements `Quantity` and `Unit` to the element `MeasurementQuantity` and the instances of the element `PhysicalDimension` related to the element `Unit`. The relations points to the elements which are copied. The instances of the child elements of the related elements are not copied, in the TestModel the instances of the elements `TestBedPart` or `Engine`.

The element Id's of the copied instances from the source are remembered in the implementation.

The arguments are inherit from the super classes.

4.11 CSVFromRule

This rule reads the data from an CSV file. The first line is the header, when the argument header line is not given, all other lines are data lines. If a line with data have more values then the header line only the values of the header line are used. The file is read with the character set **ISO-8859-1**. This rule is a **FROM-Rule**, so only the method `getValue()` is implemented.

The following table lists all the argument of this implementation. Keywords are case insensitive.



Argument Keyword	Datatype	Default	Description
Filename	String	<i>none</i>	The name of CSV File, the first line is the header, all following lines are data lines.
SeparatorChar	String	,	The separator character, the delimiting regular expression, see for detail Regular expression (p. 41).
HeaderLine	String	<i>none</i>	The header line, in case the CSV File have no header line, the separator is always comma (,).

An example of the usage of the rule is given below:

```
<rule id="readTrack" pointto="writeTrack">
  <procedure name="com.highqsoft.ascoba.CSVFromRule">
    <arguments>
      <property name="filename" value="{ARGV[1]}/>
      <property name="HeaderLine" value="Name,Description"/>
      <property name="SeparatorChar" value="\t"/>
    </arguments>
  </procedure>
</rule>
```

See also [CSVToRule](#)(p. 21)

4.12 CSVToRule

This rule writes the values into an CSV-file. The first line is the header, all other lines are data lines. The file is written with the character set **ISO-8859-1**. This rule is a **TO-Rule**, so only the method `setValue()` is implemented.

The following table lists all the argument of this implementation. Keywords are case insensitive.

Argument Keyword	Datatype	Default	Description
Filename	String	<i>none</i>	The name of CSV File, the first line is the header, all following lines are data lines.
SeparatorChar	String	,	The separator character.
WriteHeader	boolean	<i>yes</i>	Write the header line.

See also [CSVFromRule](#)(p. 20)

An example of the usage of the rule is given below:

```
<!-- Copy the instances into the CSV -->
```

```

<rule id="copyDim">
  <procedure name="com.highqsoft.ascoba.CSVToRule">
    <arguments>
      <property name="filename" value="csv_out.txt"/>
    </arguments>
  </procedure>
</rule>

```

4.13 DeleteInstancesRule

This rule will delete the selected instances together with the child instances. The rule extends **AscobaAbstractRule**(p. 15). The rule is a **TO-rule**, so only `setValue()` is implemented.

The relation pointing to the instances, which will be deleted will be removed. The Id of the instances is given as input at the method `setValue()`. The element is given as the argument `elementName`.

The element Id's of the deleted instances from the session are remembered in the implementation.

There is no transaction handling in this rule, the transaction must be started at the *TargetSession* and must be committed outside this rule.

The arguments are inherit from the super classes.

4.14 DistinctRule

This rule reduces the data set, all data set with an unique value of the given attribute value name will remain in the result data set. The first data set with the attribute value is taken. This rule is a sequence rule.

This rule is a sequence rule.

The following table lists all the argument of this implementation. Keywords are case insensitive.

Argument Keyword	Datatype	Default	Description
AttributeName	String	<i>none</i>	The name of the attribute.

4.15 ElementToRule

This rule stores the data sets as attributes in the given element of the ASAM ODS storage. The rule expect the names of the attribute to be identical. The Id's of the new created instances are stored in the cache for further usage. The relations are solved with the information in the cache. The rule search for the Name (Version) and parent instance in the storage. The value of the key attribute is used to store and find the instances in the cache not in the storage. The data will be converted from DT_STRING to the datatype of the attributes in the ASAM ODS storage. This rule is a **TO-Rule**, so only the method `setValue()` is implemented.

The following table lists all the argument of this implementation. Keywords are case insensitive.



Argument Keyword	Datatype	Default	Description
ReferenceName	String	<i>none</i>	The name of reference in the context with the OdsCache.
ElementName	String	<i>none</i>	The name of the application element.
InsertMode	String	<i>abort</i>	<p>What the do with the instances when the instances already exist in the server. Possible options are:</p> <ul style="list-style-type: none"> • abort (0) - abort the rule execution • ignore (1) - ignore the new instance use the existing • replace (2) - replace the existing with the current input.
RememberInstance	boolean	<i>yes</i>	Remember the Id of the new created instance element, the key of the instance is the name. When this argument is set to "yes" there must be an NameValueUnit available with the name of the Name attribute.
KeyAttribute	String	<i>id</i>	The name of the key attribute for searching and remembering the Id. The value is first used as check as a base attribute then as the attribute name. The base attribute is case blind, the attribute is case sensitive.

This rule use the implementation **OdsCache**(p. 13)

4.16 HandledInstancesRule

This rule returns the Id's of the handled instances of an element. The rule extends **Ascoba-AbstractRule**(p. 15). This rule is a **FROM-rule**, so only the method `getValue()` is implemented.

The arguments are inherit from the super classes. The argument `TargetSession` is only required to full file

The following example asks the handled instances of the element `Test`.

```
<rule id="getTest" pointto="copyRelTest">
  <!-- Select the element -->
  <procedure name="com.highqsoft.ascoba.HandledInstancesRule">
    <arguments>
      <property name="ElementName" value="Test"/>
      <property name="sourcesession" value="source"/>
    </arguments>
  </procedure>
</rule>
```

4.17 HQLXRule

This rule will execute a HQLX-Command, the result of the query is available at the `getValue()` method. Each time the `getValue()` is called the query is executed again.

The rule is a **FROM-rule**, so only the method `getValue()` is implemented.

The following table lists all the argument of this implementation. Keywords are case insensitive.

Argument Keyword	Datatype	Default	Description
HqlxFilename	String	<i>none</i>	The name of the hqlx file.
OdsSessionRef	String	<i>source</i>	The name of the ods session reference in the context.
SelectVariables	String[]	<i>none</i>	The names of variables which must be substituted by the values in the query. The names are separated by spaces (' ').

The following example execute the selection given in the file *selectmea.xml*

```
<rule id="selectMea" pointto="copyMea">
  <!-- Select the element -->
  <procedure name="com.highqsoft.ascoba.HQLXRule">
    <arguments>
      <property name="HqlxFilename" value="selectmea.xml"/>
      <property name="OdsSessionRef" value="source"/>
    </arguments>
  </procedure>
</rule>
```

4.18 LinkInstancesRule

This rule links the instances of the given application element in the target session according to the related instances from the source. Only the INFO-Relations are linked, the FATHER / CHILD relations are link during copy of the instances. No instances are copied, the instances in target session of the element is only linked to other existing instances in the target. The rule extends **CopyInstancesAbstractRule**(p. 16). This rule is a **TO-Rule**, so only the method `setValue()` is implemented.

The element is given as the argument `elementName`.

There is no transaction handling in this rule, the transaction must be started at the *TargetSession* and must be committed outside this rule.

An example of the usage of the rule is given below, the instances of the element Log are linked source to the target

```
<rule id="linkLog">
  <procedure name="com.highqsoft.ascoba.LinkInstancesRule">
    <arguments>
      <property name="ElementName" value="Log"/>
      <property name="sourcesession" value="source"/>
      <property name="targetsession" value="target"/>
    </arguments>
  </procedure>
</rule>
```

The arguments are inherit from the super classes.

4.19 RelatedInstancesRule

This rule will find the Ids of the related instances. This rule is a sequence rule. This rule extends rules `_com_highqsoft_ascoba_AscobaAbstractRule`.

The following table lists all the argument of this implementation. Keywords are case insensitive.

Argument Keyword	Datatype	Default	Description
RelationName	String	<i>none</i>	The name of the relation, from element to related application element.
RelatedElement-Name	String	<i>none</i>	The name of the related application element.

The element Ids of the copied instances from the source are remembered in the implementation.

The arguments are inherit from the super classes.

The following example get the instances of the element Log related to the instances of the element Test.

```
<rule id="loadLog">
  <procedure name="com.highqsoft.ascoba.RelatedInstancesRule">
    <arguments>
      <property name="ElementName" value="Test"/>
      <property name="sourcesession" value="source"/>
      <property name="targetsession" value="target"/>
      <property name="RelatedElementName" value="Log"/>
    </arguments>
  </procedure>
</rule>
```

```

        <property name="RelationName" value="Logs"/>
    </arguments>
</procedure>
</rule>

```

4.20 RelationToRule

This rule creates a relation between existing instances in the target storage, the relation may be a N to M relation. The name of the entries in the dataset must be the elementname.Id for both elements. The name of the relation in the dataset must be relation, when no relation name is given the first found relation between the two elements is used. This rule is a **TO-Rule**, so only the method `setValue()` is implemented.

The following table lists all the argument of this implementation. Keywords are case insensitive.

Argument Keyword	Datatype	Default	Description
ReferenceName	String	<i>none</i>	The name of reference in the context with the OdsCache.

This rule use the implementation **OdsCache**(p. 13)

4.21 RenameRule

This rule changes the names of the attributes in a dataset. This rule can be used in a sequence of rules.

The following table lists all the argument of this implementation. Keywords are case insensitive.

Argument Keyword	Datatype	Default	Description
oldNames	String[]	<i>none</i>	The old names of attributes. The names are separated by spaces (' ').
newNames	String[]	<i>none</i>	The new names of attributes. The names are separated by spaces (' '). The length of the list of new names must be exactly the same as the length of the list with the old names. The order in the list of the old names and new names are important.

4.22 SetSystemPropertyRule

This rule stores the first found attribute value as a system property. The system property can be used by the external calling java program, such as ant.

This rule is a sequence rule.

The following table lists all the argument of this implementation. Keywords are case insensitive.

Argument Keyword	Datatype	Default	Description
AttributeName	String	<i>none</i>	The name of the attribute.
PropertyName	String	<i>Name of the attribute</i>	The name of the property where the first found attribute value is stored. The default is the name of the attribute.

4.23 SetUndefinedRule

This rule set the value of an attribute to undefined (flag = 0) when the value of the attribute match with the given attribute value. The values for all attributes of the data set are check and set to undefined when required.

The DataType of the value must be DT_STRNG. There is a case sensitive string compare.

This rule is a sequence rule.

The following table lists all the argument of this implementation. Keywords are case insensitive.

Argument Keyword	Datatype	Default	Description
AttributeValue	String	<i>none</i>	The value of the attribute.

4.24 StoreRule

This rule save to or restored from the data set the context. When the rule is used as a FROM rule the data set is restored from the context returned at getValue. When the rule is used as a TO rule the data set given at setValue is stored in the context with a given name.

When the rule is used as the first or last rule in a ruleseq the data set is saved or restored from the context.

The following table lists all the argument of this implementation. Keywords are case insensitive.

Argument Keyword	Datatype	Default	Description
ReferenceName	String	<i>none</i>	The name of the reference to store or restore the data.

The following example stores or restores the data set with the name StoreNVU in the context.

```
<rule id="Store">
  <procedure name="com.highqsoft.ascoba.StortRule">
    <arguments>
      <property name="ReferenceName" value="StoreNVU"/>
    </arguments>
  </procedure>
</rule>
```

4.25 WriteAttributeRule

This rule writes the attribute value to a file, each line is an attribute value.

This rule is a sequence rule.

The following table lists all the argument of this implementation. Keywords are case insensitive.

Argument Keyword	Datatype	Default	Description
AttributeName	String	<i>none</i>	The name of the attribute.
HeaderLine	String	<i>none</i>	The header line of the file.
PrefixLine	String	<i>none</i>	The prefix of the line with the attribute value.
PostfixLine	String	<i>none</i>	The postfix of the line with the attribute value.
FooterLine	String	<i>none</i>	The footer line of the file.
Filename	String	<i>name of attribute</i>	The name of the file, the default is the name of the attribute with the extension ".txt".

4.26 XLSDate2ODSRule

This rule converts the numeric value from an XLS Date to an ODS Date string. The numeric value 0 is 1.1.1900. the fraction of the value is the fraction of the day. This rule is a sequence rule.

This rule is a sequence rule.

The following table lists all the argument of this implementation. Keywords are case insensitive.

Argument Keyword	Datatype	Default	Description
AttributeName	String	<i>none</i>	The name of the attribute.

4.27 XLSFromRule \addindex XLS \addindex from rule XLS

This rule reads the data from an XLS (Microsoft Excel (97 - 2000) file. Thre rule reads the header line and expect string values in the header line which will ne used as names of the fields in the data set. The rule reads the given numner of rows from each line in the XLS file. The rule reads the values from the file beginning with first line with values and skip the empty lines until the number of continious empty lines are reached. This rule reads only cells with string or numeric values, all values are converted to string values. This rule is a **FROM-Rule**, so only the method `getValue()` is implemented.

The following table lists all the argument of this implementation. Keywords are case insensitive.



Argument Keyword	Datatype	Default	Description
Filename	String	<i>none</i>	The name of XLS file
Sheet	String	<i>Table1</i>	The name of XLS sheet in the file
HeaderLine	String	<i>none</i>	<i>Optinal, not used when HeaderLine-Number is given.</i> The header line, in case the XLS file have no header line, the separator is always comma (.).
HeaderLineNumber	Integer	<i>none</i>	The number of the header line in the XLS file. The first line in the file starts with the index 1
FirstRow	Integer	<i>1</i>	The number of first rows to read from each line out of XLS file. The first rows in the file starts with the index 1
NumberOfRows	Integer	<i>none</i>	The number of rows to read from each line out of XLS file. When this argument is not given, the number of rows match the number of used cells in the header line or the number of names in the header line
FirstValueLine	Integer	<i>none</i>	The number of first line in the XLS file. The first line in the file starts with the index 1
NumberEmptyLines	Integer	<i>1</i>	The number of empty lines before stop to read from the XLS file.
DateColumns	String	<i>none</i>	The names of the columns which must be read from the XLS-File as a date. The different columns are separated with comma (,)

An example of the usage of the rule is given below:

```
<rule id="readPhysical_Unit">
  <procedure name="com.highqsoft.ascoba.XLSFromRule">
    <arguments>
      <property name="filename" value="{XLS.Source}"/>
      <property name="sheet" value="Physical_Unit"/>
      <property name="numberOfRows" value="9"/>
      <property name="headerLineNumber" value="1"/>
      <property name="firstValueLine" value="2"/>
    </arguments>
  </procedure>
</rule>
```

Chapter 5

The TestModel

5.1 The application model of the test.

The application model used for the test of ascoba and in the documentation of the rules is given below.

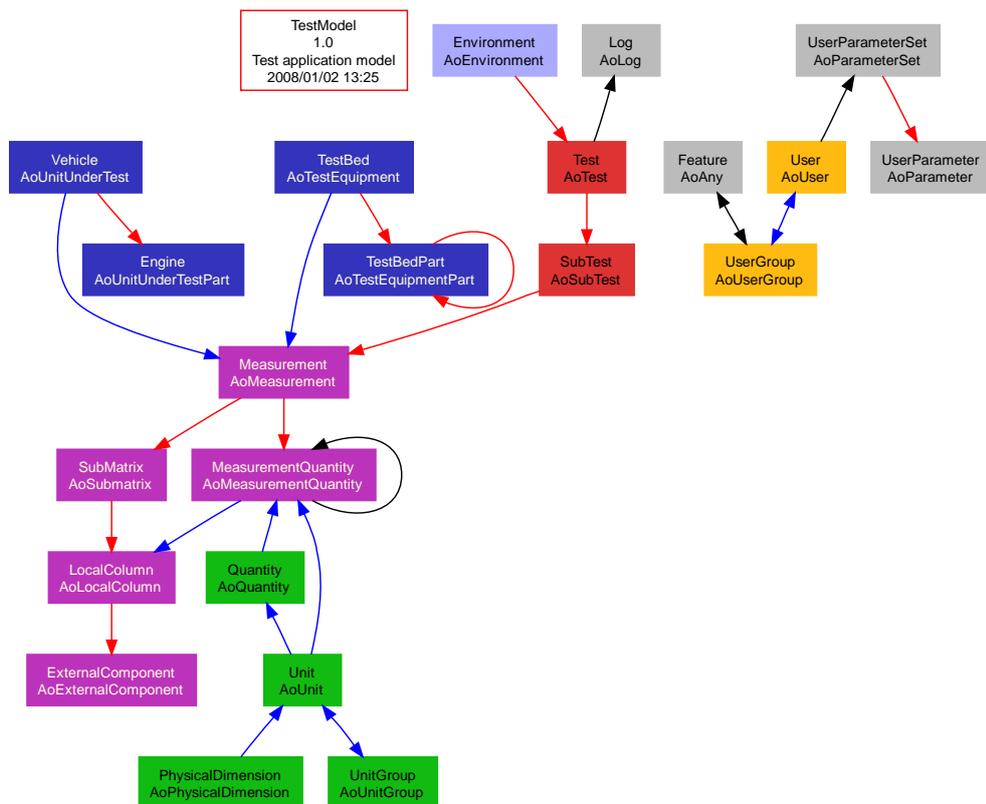


Figure 5.1: The TestModel



Chapter 6

Example files

6.1 The Athos INI-File.

```
; Athos initialization and configuration file.
;
; Syntax conventions:
;
; - Semicolon is the comment sign.
; - String may optionally be enclosed in double quotes. If the double
;   quotes are omitted, leading and trailing whitespaces are removed.
; - German Umlauts (mutated vowels) and non-printable characters
;   follow the Asam Transport Format (ATF) conventions for maximum
;   portability in hetergenious environments.
; - Only one entry per line is allowed.
; - Variables may be constructed by using other variables. Substitutions
;   are indicated by $(your_variable_name) (similar to Unix make).
; - Beware of recursive definitions. No checks are in the code yet.
;   Recursive substitutions will result in an infinite memory-sucking loop!!!

[ATHOS]                                ; Athos system definitions.
ATHOS_ROOT      = "/users/xenon1/conv/athos"      ; Fallback if not in environment.
BASE_MODEL_URL  = "file:///$(ATHOS_ROOT)/etc/ao_base51.htm"
DEBUGLEVEL     = 0
MAX_LC_MEMORY  = 10
MAX_NUMBER_LC  = 500
FREE_NUMBER_LC = 300
ASCOBA_LOGFILE = "file:///$(ATHOS_ROOT)/log/ascoba.log"
MAX_RUN_INST   = 1000
LOG_MAX_LINES  = 000000
AOP_SERVICE    = "TestModel/DB"

[SERVICE "XATF/File"]
DRIVER      = xatf
DIRECTORY   = "file:///$(ATHOS_ROOT)/bin/$(OSTYPE)/"
NOSECURITYACTIVE = YES
IGNORE_SECURITY = YES
SEARCH_FOR_BASE_REF = NO
REOPEN_VARIABLE = FILENAME
XATF_LOGFILE = "file:///$(ATHOS_ROOT)/log/xatf_file.log"
ODS_LOGFILE = "file:///$(ATHOS_ROOT)/log/xatf_file_ods.log"
ODSVERSION = "5.0 readonly"
LOAD_NEXT_ID = NO
;WRITE_MODE=CHANNEL
RUN_SINGLETHREADED = YES
XATF_WRITE_SECURITY = YES
```

```
LOG_EVENTS = YES
```

6.2 An ASCOBA example file.

An example of an ascoba command is given below, the used implementations and rules are given above. ASCOBA can be started with the following command, which will copy the instance of the element *Measurement* with the Id 65: `java -jar ascoba2g.jar ascoba_example.xml test.atfx Measurement 65`

6.2.1 The ascoba.xml file.

This file is the MoMo XML-Control file.

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

<job xmlns="http://www.highqsoft.de/schema" gathermode="PEDANTIC" messagelevel="NONE" tracelevel="NONE" loglevel="STATUS"

  <environment name="env"/>

  <properties>
    <property name="root.dir" location="${env.PROJECTS_ROOT}/highqsoft/athos/ascoba"/>
    <property name="ATHOS_INI" location="${env.PROJECTS_ROOT}/highqsoft/athos/ascoba/etc/ascoba.ini"/>
    <property name="SelectElementMea" value="Measurement"/>
    <property name="InstanceMeaId" value="11"/>
    <property name="SelectElementDim" value="PhysicalDimension"/>
    <property name="SelectElementUsr" value="User"/>
    <property name="SelectDIM" location="${env.PROJECTS_ROOT}/highqsoft/athos/ascoba/etc/selectdim.xml"/>
    <property name="SelectMEA" location="${env.PROJECTS_ROOT}/highqsoft/athos/ascoba/etc/selectmea.xml"/>
    <property name="SelectUSR" location="${env.PROJECTS_ROOT}/highqsoft/athos/ascoba/etc/selectusr.xml"/>
    <property name="WithSecurity" value="false"/>
  </properties>

  <logger name="com.highqsoft.xsd.DefaultHandler">
    <formatter name="com.highqsoft.xsd.DefaultFormatter">
      <arguments>
        <property name="FORMAT" value="{0,date} {0,time}. {1,number,###} diff: {2,number,####} Thread 0x{3} {6} {4}>
        <property name="LINEBREAK" value="&#13;&#9;"/>
        <property name="SEPARATOR" value="&#13;*****"/>
      </arguments>
    </formatter>
  </logger>

  <messenger name="com.highqsoft.xsd.DefaultHandler">
    <formatter name="com.highqsoft.xsd.DefaultFormatter">
      <arguments>
        <property name="FORMAT" value="{0,date} {0,time}. {1,number,###} diff: {2,number,####} Thread 0x{3} {6} {4}>
        <property name="LINEBREAK" value="&#13;&#9;"/>
        <property name="SEPARATOR" value="&#13;*****"/>
      </arguments>
    </formatter>
  </messenger>

  <tracer name="com.highqsoft.xsd.DefaultHandler">
    <formatter name="com.highqsoft.xsd.DefaultFormatter">
      <arguments>
        <property name="FORMAT" value="{0,date} {0,time}. {1,number,###} diff: {2,number,####} Thread 0x{3} {6} {4}>
        <property name="LINEBREAK" value="&#13;&#9;"/>
        <property name="SEPARATOR" value="&#13;*****"/>
      </arguments>
    </formatter>
  </tracer>
```

```

<init>
  <source>
    <aosession user="karst" password="secret" id="source">
      <aofactory servicename="TestModel/DB">
        <aoservice plugablename="CORBA" nameservicehost="TestServer" nameserviceport="2809"/>
      </aofactory>
    </aosession>
  </source>

  <target>
    <aosession filename="{ARGV[1]}" openmode="w" id="target">
      <aofactory servicename="XATF/File">
        <aoservice plugablename="ATHOS" inifile="{ATHOS_INI}"/>
      </aofactory>
    </aosession>
  </target>

  <terminate>
    <implementation name="com.highqsoft.ascoba.CopyModel">
      <arguments>
        <property name="withsecurity" value="{WithSecurity}"/>
      </arguments>
    </implementation>
  </terminate>

</init>
<!-- Definition of the rules -->
<ruledef drivenby="FROM">
  <fromrules>
    <ruleseq>
      <!-- Select the element AoPhysicalDimension -->
      <rule id="selectDim" pointto="copyDim">
        <procedure name="com.highqsoft.ascoba.HQLXRule">
          <arguments>
            <property name="HqlxFilename" value="{SelectDIM}"/>
            <property name="OdsSessionRef" value="source"/>
          </arguments>
        </procedure>
      </rule>

      <!-- Get the instance of AoMeasurement -->
      <rule id="getMea" pointto="copyMea">
        <procedure name="com.highqsoft.ascoba.AscobaInputRule">
          <arguments>
            <property name="ElementName" value="{SelectElementMea}"/>
            <property name="InstanceId" value="{InstanceMeaId}"/>
            <property name="sourcesession" value="source"/>
          </arguments>
        </procedure>
      </rule>

      <!-- Select the instance of element AoUser -->
      <rule id="selectUsr" pointto="copyUsr">
        <procedure name="com.highqsoft.ascoba.HQLXRule">
          <arguments>
            <property name="HqlxFilename" value="{SelectUSR}"/>
            <property name="OdsSessionRef" value="source"/>
          </arguments>
        </procedure>
      </rule>

      <!-- Get the Ids of handled instances of element Test -->
      <rule id="getTest" pointto="copyRelTest">
        <procedure name="com.highqsoft.ascoba.HandledInstancesRule">
          <arguments>
            <property name="ElementName" value="Test"/>
          </arguments>
        </procedure>
      </rule>
    </ruleseq>
  </fromrules>
</ruledef>

```

```

        </arguments>
    </procedure>
</rule>
<!-- Get the Ids of handled instances of element TestBed -->
<rule id="getTestBed" pointto="copyTestBedPart">
    <procedure name="com.highqsoft.ascoba.HandledInstancesRule">
        <arguments>
            <property name="ElementName" value="TestBed"/>
        </arguments>
    </procedure>
</rule>
</ruleseq>
</fromrules>
<torules>
<ruleseq>
    <!-- Copy the instances of the AoPhysicalDimension into the target. -->
    <rule id="copyDim" sessionref="target">
        <headtransaction type="START"/>
        <procedure name="com.highqsoft.ascoba.CopyInstancesRule">
            <arguments>
                <property name="ElementName" value="${SelectElementDim}"/>
                <property name="withsecurity" value="${WithSecurity}"/>
            </arguments>
        </procedure>
    </rule>
    <!-- Copy the instances of the AoMeasurement, with the related instances into the target. -->
    <rule id="copyMea" sessionref="target">
        <procedure name="com.highqsoft.ascoba.CopyInstancesWithRelatedRule">
            <arguments>
                <property name="ElementName" value="${SelectElementMea}"/>
                <property name="withsecurity" value="${WithSecurity}"/>
            </arguments>
        </procedure>
    </rule>
    <!-- Copy the instances of the AoUser into the target. -->
    <rule id="copyUsr" sessionref="target">
        <procedure name="com.highqsoft.ascoba.CopyInstancesRule">
            <arguments>
                <property name="ElementName" value="${SelectElementUsr}"/>
                <property name="withsecurity" value="${WithSecurity}"/>
            </arguments>
        </procedure>
    </rule>
    <!-- Get the instances of element Log related to the instance of the Element Test and
        Copy the instances of the Element Log into the target. -->
    <rule id="copyRelTest" sessionref="target">
        <ruleseq>
            <rule id="loadLog">
                <procedure name="com.highqsoft.ascoba.RelatedInstancesRule">
                    <arguments>
                        <property name="ElementName" value="Test"/>
                        <property name="RelatedElementName" value="Log"/>
                        <property name="RelationName" value="Logs"/>
                    </arguments>
                </procedure>
            </rule>
            <rule id="copyLog">
                <procedure name="com.highqsoft.ascoba.CopyInstancesRule">
                    <arguments>
                        <property name="ElementName" value="Log"/>
                        <property name="withsecurity" value="${WithSecurity}Y"/>
                    </arguments>
                </procedure>
            </rule>
        </ruleseq>
    </rule>
    <!-- Copy the instances of the AoUser into the target. -->

```

```

    <rule id="copyTestBedPart" sessionref="target">
      <procedure name="com.highqsoft.ascoba.CopyInstancesChildrenRule">
        <arguments>
          <property name="ElementName" value="TestBed"/>
          <property name="withsecurity" value="{WithSecurity}Y"/>
        </arguments>
      </procedure>
      <tailtransaction type="COMMIT"/>
    </rule>
  </ruleseq>
</torules>
</ruledef>
</job>

```

6.2.2 The file selectdim.xml

The file to select the instances of AoPhysicalDimension

```

<?xml version="1.0" encoding="UTF-8"?>

<!--
  Select the Id's of all instances of AoPhysicalDimension so they can be copied to ATF/XML file.
  Used by ascoba.

  Modification history:

  $Log: selectdim.xml,v $
  Revision 1.2  2008/01/07 15:49:13  karst
  More details implemented.

  Revision 1.1  2007/11/21 08:33:06  karst
  Initial version

-->
<hqlx xmlns="http://www.highqsoft.de/schema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.highqsoft.de/schema http://www.highqsoft.de/schema" >

  <querystructureext>

    <!--
      The Attribute list to be selected.
    -->

    <anuseq>
      <!-- The Id of the AoPhysicalDimension -->
      <selaidnameunitid>
        <aidname basename="id">
          <aid basename="AoPhysicalDimension"/>
        </aidname>
      </selaidnameunitid>
    </anuseq>

    <!--
      The conditions.
    -->

    <condseq>
      <!-- The condition:
      AoPhysicalDimension.Name == "*"
    -->
      <selitem>
        <selvalueext selopcode="LIKE">
          <aidnameunitid>
            <aidname basename="name">
              <aid basename="AoPhysicalDimension"/>
            </aidname>
          </aidnameunitid>
        </selvalueext>
      </selitem>
    </condseq>
  </querystructureext>
</hqlx>

```

```

        </aidname>
        </aidnameunitid>
        <ts_value value="*" datatype="DT_STRING"/>
    </selvalueext>
</selitem>

</condseq>

</querystructureext>
</hqlx>

```

6.2.3 The file selectusr.xml

The file to select the instances of AoUser.

```

<?xml version="1.0" encoding="UTF-8"?>
<!--
Select an Id of an instance element of AoUser to be copied to the ATF/XML file.
Used by ascoba.

Modification history:

$Log: selectusr.xml,v $
Revision 1.2  2007/12/20 08:06:50  karst
Select all users with an a.

Revision 1.1  2007/11/27 09:05:55  karst
Initial CVS version.

Revision 1.1  2007/11/21 08:33:06  karst
Initial version

-->
<hqlx xmlns="http://www.highqsoft.de/schema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.highqsoft.de/schema http://www.highqsoft.de/schema/highqsoft.xsd" >
  <querystructureext>
    <!--
    The Attribute list to be selected.
    -->
    <anuseq>
      <!-- The Id of the AoUser -->
      <selaidnameunitid>
        <aidname basename="id">
          <aid basename="AoUser"/>
        </aidname>
      </selaidnameunitid>
    </anuseq>
    <!--
    The conditions.
    -->
    <condseq>
      <!-- The condition:
      AoUser.Name == "*a*"
      -->
      <selitem>
        <selvalueext selopcode="LIKE">
          <aidnameunitid>
            <aidname basename="name">
              <aid basename="AoUser"/>
            </aidname>
          </aidnameunitid>
        </selvalueext>
      </selitem>
    </condseq>
  </querystructureext>
</hqlx>

```

```
        </aidname>
        </aidnameunitid>
        <ts_value value="*a*" datatype="DT_STRING"/>
    </selvalueext>
</selitem>

</condseq>

</querystructureext>
</hqlx>
```



Chapter 7

Appendix

7.1 Regular expression

Summary of regular-expression constructs

Characters

Construct	Matches
<code>x</code>	The character <code>x</code>
<code>\\</code>	The backslash character
<code>\0n</code>	The character with octal value <code>0n</code> ($0 \leq n \leq 7$)
<code>\0mn</code>	The character with octal value <code>0mn</code> ($0 \leq n \leq 7$)
<code>\0mnn</code>	The character with octal value <code>0mnn</code> ($0 \leq m \leq 3, 0 \leq n \leq 7$)
<code>\xhh</code>	The character with hexadecimal value <code>0xhh</code>
<code>\uhhhh</code>	The character with hexadecimal value <code>0xhhhh</code>
<code>\t</code>	The tab character (<code>'\u0009'</code>)
<code>\n</code>	The newline (line feed) character (<code>'\u000A'</code>)
<code>\r</code>	The carriage-return character (<code>'\u000D'</code>)
<code>\f</code>	The form-feed character (<code>'\u000C'</code>)
<code>\a</code>	The alert (bell) character (<code>'\u0007'</code>)
<code>\e</code>	The escape character (<code>'\u001B'</code>)
<code>\cx</code>	The control character corresponding to <code>x</code>

Character classes

Construct	Matches
[abc]	a, b, or c (simple class)
[^abc]	Any character except a, b, or c (negation)
[a-zA-Z]	a through z or A through Z, inclusive (range)
[a-d[m-p]]	a through d, or m through p: [a-dm-p] (union)
[a-z&&[def]]	d, e, or f (intersection)
[a-z&&[^bc]]	a through z, except for b and c: [ad-z] (subtraction)
[a-z&&[^m-p]]	a through z, and not m through p: [a-lq-z](subtraction)

Predefined character classes

Construct	Matches
.	Any character (may or may not match line terminators)
\d	A digit: [0-9]
\D	A non-digit: [^0-9]
\s	A whitespace character: [\t\n\r\x0B\f]
\S	A non-whitespace character: [^\s]
\w	A word character: [a-zA-Z_0-9]
\W	A non-word character: [^\w]

Chapter 8

Modification history

2008/01/07

Initial version

2008/02/01

Add **Regular expression**(p. [41](#))

Index

- add literal, [15](#), [26](#)
- application model compare, [11](#)
- application model copy, [11](#)

- children copy instances, [18](#)
- compare application model, [11](#)
- compare model, [11](#)
- copy application model, [11](#)
- copy instances, [18](#), [19](#)
- copy instances children, [18](#)
- copy instances with father and children, [18](#)
- copy instances with father, children and info-to related, [19](#)
- copy instances with info-to related, [19](#)
- copy model, [11](#)
- copy security, [11](#)
- copy security, [17](#)
- csv, [20](#), [21](#)

- data set distinct, [22](#)
- dataset, [8](#)
- delete instances, [22](#)
- distinct data set, [22](#)

- element to ods, [22](#)

- father and children copy instances, [18](#)
- father copy instances, [18](#)
- from rule csv, [20](#)

- getValue(), [8](#)

- handled instances, [24](#)
- hqlx, [24](#)

- input instance, [16](#)
- instance input, [16](#)
- instances copy, [18](#), [19](#)
- instances copy children, [18](#)
- instances copy with father and children, [18](#)
- instances copy with father, children and info-to related, [19](#)
- instances copy with info-to related, [19](#)
- instances delete, [22](#)
- instances link, [25](#)

- link instances, [25](#)

- literal add, [15](#), [26](#)

- mail, [4](#)
- model compare, [11](#)
- model copy, [11](#)

- NameValueUnit, [8](#)

- object definition sequence , [4](#)

- query, [24](#)

- regular expression, [41](#)
- related copy instances, [19](#)
- related instances , [25](#)
- relation to ods, [26](#)
- restore data set, [27](#)
- rule, [8](#)

- security copy, [11](#), [17](#)
- setValue(), [8](#)
- store data set , [27](#)

- to rule, [22](#), [26](#)
- to rule csv, [21](#)

- XLS date to ODS, [27](#), [28](#)