



ARIS Risk & Compliance Manager **CUSTOMIZING GUIDE**

Version 10.0 - Service Release 3

December 2017

Document content not changed since release 10.0.2. It applies to version 10.0.3 without changes.

This document applies to ARIS Risk & Compliance Manager Version 10.0 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 2010 - 2017 Software AG, Darmstadt, Germany and/or Software AG USA Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <http://softwareag.com/licenses>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

Contents

1	Text conventions	1
2	What can be customized?.....	2
3	General procedure	3
3.1	Adapt the XML configuration	3
3.2	Adapt rules.....	3
3.3	Adapt names	3
3.4	Inheritance.....	4
3.4.1	Inheritance hierarchy of central objects.....	4
3.4.2	Object and VersionObject object types	4
3.4.3	TransactionalObject object type.....	5
3.4.4	MonitorableObject object type.....	6
3.4.5	RecurringObject object type.....	7
3.4.6	ObjectContainer object type.....	8
3.4.7	Inheritance in the file objectTypes.xml	9
3.5	Conventions.....	11
3.5.1	Conventions in the XML configuration.....	11
3.5.2	Conventions for object generation	11
3.5.2.1	Environment association in environment-specific objects.....	11
3.5.2.2	MonitorableObject object type.....	11
3.5.2.3	Identical attribute names	12
3.5.2.4	Object assignment if names are identical	12
3.6	Class mappings	13
3.6.1	Actions	13
3.6.2	Command class mappings	14
3.6.3	Statistics class mappings.....	15
3.6.4	BI class mappings.....	15
3.6.5	UI class mappings	16
3.6.6	View class mappings.....	16
3.6.7	VCREG.XML configuration file	17
3.7	Customize help	17
4	Basic use cases	18
4.1	Customize object properties.....	18
4.1.1	Overwrite the schema version.....	18
4.1.2	Add/adapt a simple attribute.....	19
4.1.2.1	Create a simple attribute.....	19
4.1.2.1.1	Adapt an object type	19
4.1.2.1.2	Add/adapt properties.....	19
4.1.2.1.3	Assign validator	20
4.1.2.1.4	Assign converter.....	20
4.1.2.2	Add an attribute to a form	21
4.1.2.2.1	Adapt a form.....	21
4.1.2.2.2	Add/adapt properties of a form	21
4.1.2.2.3	Assign a renderer	22
4.1.2.2.4	Adapt rules	22
4.1.2.2.5	Add/adapt reports.....	22
4.1.2.3	Add an attribute to a list	23
4.1.2.3.1	Adapt a list	23
4.1.2.3.2	Add/adapt properties of a list	23
4.1.2.3.3	Adapt data retrieval for a list	24

4.1.2.3.4	Add a renderer	24
4.1.2.3.5	Add/adapt reports.....	25
4.1.2.4	Add an attribute to a filter	25
4.1.2.4.1	Adapt a list filter	25
4.1.2.4.2	Add/adapt properties of a filter	25
4.1.2.4.3	Assign a renderer	26
4.1.3	Add/modify an enumeration attribute.....	26
4.1.3.1	Create an enumeration attribute.....	26
4.1.3.1.1	Add/adapt an enumeration.....	26
4.1.3.1.2	Add/adapt properties of an enumeration	26
4.1.3.1.3	Adapt an object type	27
4.1.3.1.4	Add an attribute to a form.....	27
4.1.3.1.5	Adapt data retrieval for a list	27
4.1.3.1.6	Add an attribute to a list	27
4.1.3.1.7	Add an attribute to a filter.....	27
4.1.4	Add/adapt a list attribute	28
4.1.4.1	Create a list attribute.....	28
4.1.4.1.1	Adapt an object type	28
4.1.4.1.2	Add/adapt properties.....	28
4.1.4.1.3	Adapt list restrictions.....	29
4.1.4.1.4	Adapt roles	30
4.1.4.1.5	Add an attribute to a form.....	31
4.1.4.2	Add a selection list	32
4.1.4.2.1	Adapt a selection list	32
4.1.4.2.2	Add/adapt properties.....	32
4.1.4.2.3	Adapt data query for selection list	33
4.1.4.2.4	Assign a renderer	33
4.1.4.2.5	Add a selection list filter.....	33
4.2	Customize the object life cycle.....	34
4.2.1	Workflow configuration	34
4.2.1.1	Add a state	34
4.2.1.1.1	Add a state to an active object	35
4.2.1.1.2	Add a state to a deleted object	35
4.2.1.2	Add a transition	36
4.2.1.2.1	Add a prepare transition	36
4.2.1.2.2	Add an insert transition	37
4.2.1.2.3	Add an update transition.....	38
4.2.1.2.4	Add a reset transition	39
4.2.1.2.5	Add a delete transition	40
4.2.1.2.6	Add a recover transition.....	41
4.2.2	Configure the command chain catalog	42
4.2.2.1	Modify a command chain.....	42
4.2.2.2	Add a command chain.....	43
4.2.3	Adapt/add user interactions	44
4.2.3.1	Confirmation dialogs.....	44
4.2.3.2	Input dialogs	45
4.3	Adapt the task configuration	48
4.4	Adapt a master data import	51
4.5	Add/adapt hierarchies	52
4.5.1	Add an enumeration item	52
4.5.2	Add a new list element to a master data object	53
4.5.3	Add a new list element to a transactional object	54

4.5.4	Display and input options for forms.....	54
4.5.5	Automatic transfer of hierarchy objects	55
4.5.6	Make a hierarchy attribute editable.....	55
4.5.7	Assign roles to a hierarchy attribute.....	55
4.5.8	Add a hierarchy evaluation	55
4.5.9	Create a new data view for hierarchy statistics	56
4.6	Add/adapt statistics	57
4.6.1	Adapt statistics	57
4.6.1.1	Adapt column widths	57
4.6.1.2	Link structural elements.....	58
4.6.1.3	Add/adapt columns	59
4.6.1.3.1	statistic.columnGroup.enum-based statistics.....	59
4.6.1.3.2	statistic.columnGroup.perCent-based statistics.....	60
4.6.1.3.3	statistic.column.value-based statistics	61
4.6.1.3.4	Adapt links	62
4.6.1.3.5	Use a new hierarchy.....	62
4.7	Add/adapt reports	63
4.7.1	Add/adapt reports for forms	63
4.7.1.1	Replace an existing form report definition	63
4.7.1.2	Add a new form report definition	64
4.7.1.3	Incorporate a new form report selection	64
4.7.2	Add/adapt reports for lists.....	65
4.7.2.1	Replace an existing list report definition	65
4.7.2.2	Add a new list report definition.....	65
4.7.2.3	Incorporate a new report selection.....	66
4.8	Modify message template	67
4.8.1	Add a new message template	67
4.8.2	Add a new message template content	67
4.8.3	Customize the contents of a message template	68
4.8.4	Send messages.....	69
4.9	Add/adapt segregation of duties	70
4.10	Add/adapt rule.....	71
4.10.1	Overwrite an existing rule file	71
4.10.2	Incorporate a new rule file.....	72
4.10.3	Reuse existing rules for new attributes	73
4.11	Add/adapt a scheduled task	74
4.11.1	Adapt the schedule.....	74
4.11.2	Generator	76
4.11.2.1	Adapt the object search	76
4.11.2.2	Generate objects.....	77
4.11.3	Adapt the object search	78
4.11.4	Updater	78
4.12	Adapt offline processing	78
4.12.1	Modify offline documents.....	79
4.12.2	Change the offline operator roles definition	80
4.12.3	Add a new Offline editor role.....	80
4.12.4	Adapt offline processors	81
4.12.5	Adapt offline behavior for each object type	81
4.13	Add/adapt dashboard link	82
4.13.1	Adapt DashBoard link	82
4.13.2	Add dashboard link.....	83
4.13.2.1	Add a MashZone list for object data	83

4.13.2.2	Add a MashZone list for object links	84
4.13.2.3	Assign a name to a MashZone list.....	84
4.14	Adjust navigation	85
4.14.1	Adapt navigation for an area.....	85
4.15	Adapt and extend event enabling.....	87
4.15.1	Extend existing event type XSDs	87
4.15.2	Create new event type XSDs.....	88
4.16	Adapt interface appearance.....	88
4.16.1	Exchange images and icons	88
4.16.2	Include CSS files	89
4.16.3	Include JavaScript files	89
5	Disclaimer.....	90
6	Software AG support	91

1 Text conventions

Menu items, file names, etc. are indicated in texts as follows:

- Menu items, keyboard shortcuts, dialogs, file names, entries, etc. are shown in **bold**.
- Content input that you specify is shown in **<bold and within angle brackets>**.
- Single-line example texts are separated at the end of a line by the character ↵, e.g., a long directory path that comprises multiple lines.
- File extracts are shown in the following font:

`This paragraph contains a file extract.`

2 What can be customized?

The configuration of ARIS Risk & Compliance Manager is defined in XML files based on commented XML schema files (.xsd). These XML files describe:

- characteristics of objects and attributes as well as their representation in the user interface
- object life cycle and form flow
- Hierarchies
- statistics and reports
- roles, privileges, and segregation of duties
- notifications
- scheduled tasks
- Offline processing
- Dashboard links
- Style sheets

The XML configuration refers to:

- Java classes that implement specific behavior
- property files containing the localized text for the user interface
- DRL and DSL files containing the rules for the forms

You can customize the application by adapting the elements in the XML configuration.

For example, you can add an existing attribute type to an existing object type using a new name. Or you can include new elements, such as new messages or Java classes implementing new behavior that you cannot otherwise configure using XML. This does not require an individual build process. Changes to the configuration are applied during server runtime as long as the system is running in development mode (ACC parameter **arcm.config.isDevelopmentSystem = true**). A server restart is required for productive systems. The section **Basic use cases** (Page 18) describes all steps for adaptations based on the XML configuration.

You can also customize the behavior by integrating individual Java implementations of special interfaces through the XML configuration. These Java classes can be individually developed based on a defined interface and are then included like the default classes without any particular build process being required for ARIS Risk & Compliance Manager.

If an updated version of this document is available, you will find it here:

<http://aris.softwareag.com/ARISDownloadCenter/ADCDocumentationServer>
(<http://aris.softwareag.com/ARISDownloadCenter/ADCDocumentationServer>)

3 General procedure

The default implementation is based on configuration mechanisms that are also used for adaptations. The default configuration of the control-based approach (CBA) is already an adaptation of the default configuration of the risk-based approach (RBA). If the default behavior or default structure is to be changed selectively, the corresponding passages in the XML configuration can be overwritten so that a changed behavior or a changed or extended structure is defined at this position. These selective changes and extensions are performed in the directory **tomcat\webapps\arc\WEB-INF\config\custom**, which is located in the installation directory.

The chapter on **Basic use cases** (Page 18) describes the steps necessary to create adapted configurations based on the use cases supported. The procedure described is implemented in configuration examples that you can find in the folder **customizing examples**.

The folder **standard configuration** contains the default XML configuration files. You can use these files as a starting point for adaptations and copy the passages to be customized from there and then change them according to your requirements.

3.1 Adapt the XML configuration

The **xml** folder contains one or more XML files including the customized XML configuration. The system validates these files against the XML schema file **custom.xsd**. This file is located in the **xsd** folder and must not be changed. The **<custom>** element must be the root element of these XML files.

3.2 Adapt rules

The **rules** folder contains adapted rule files that are integrated by the XML configuration. See **Add/adapt rule** (Page 71).

3.3 Adapt names

The **properties** folder contains one or more property files with customized strings for the user interface.

Naming conventions:

- The file name must end with **_xx.properties**. (**xx** stands for the code of the language the strings are localized in.)
- The underscore (**_**) must not occur at any other position of the file name.

Example

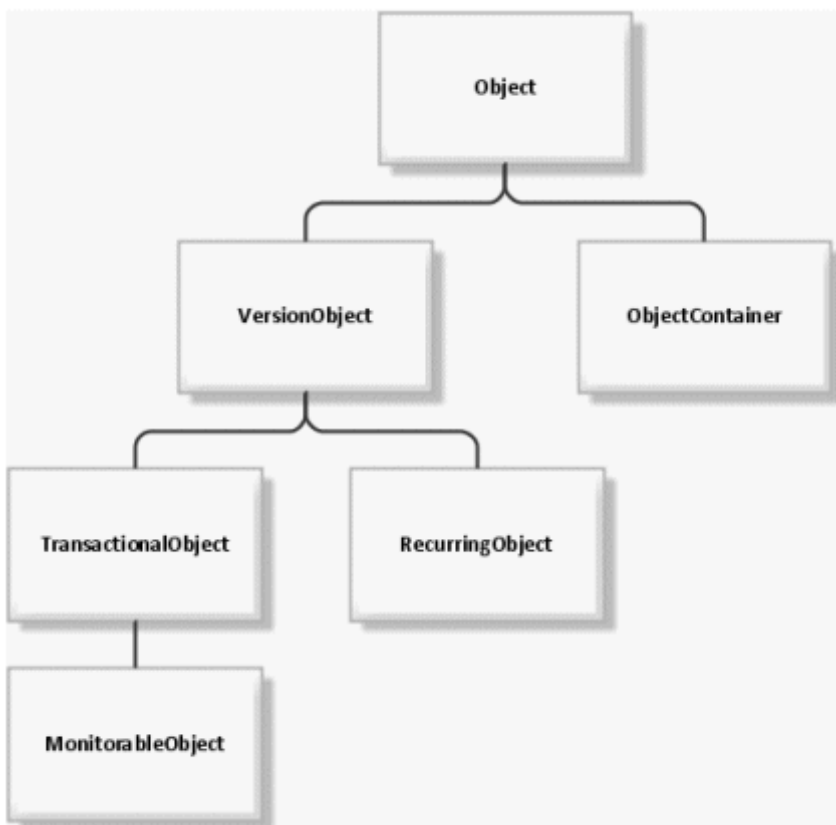
For English, you can use **myCustomizedStrings_en.properties**, but not **my_customized_strings_en.properties**.

This allows you to create several language versions in parallel, whose file names only differ in terms of the language code.

3.4 Inheritance

ARIS Risk & Compliance Manager version 4.0 introduces an inheritance mechanism in object configuration (**objectTypes.xml**). This ensures a uniform structure of objects and attributes of the components, Test Management, Issue Management, etc., as well as objects with similar meaning and function. The inheritance determines the function of an object type within the component and thus reduces the configuration effort. The configuration effort is reduced due to the centralization and reuse of workflow-relevant attributes. The inheritance feature also simplifies the programming of generic system functions, such as monitoring because it is possible to access centrally configured attributes.

3.4.1 Inheritance hierarchy of central objects



3.4.2 Object and VersionObject object types

The **Object** and **VersionObject** object types (**objectTypes.xml**: **OBJECT**, **VERSION**) contain central technical attributes. These should not be changed during customizing. Object types that are subject to the ARIS Risk & Compliance Manager versioning mechanism need to extend the **VersionObject** object type. Non-versioned object types inherit from **Object** directly. Similar to the Java programming language, it is not necessary to explicitly specify the extension of the **Object** object type, it is extended automatically.

3.4.3 TransactionalObject object type

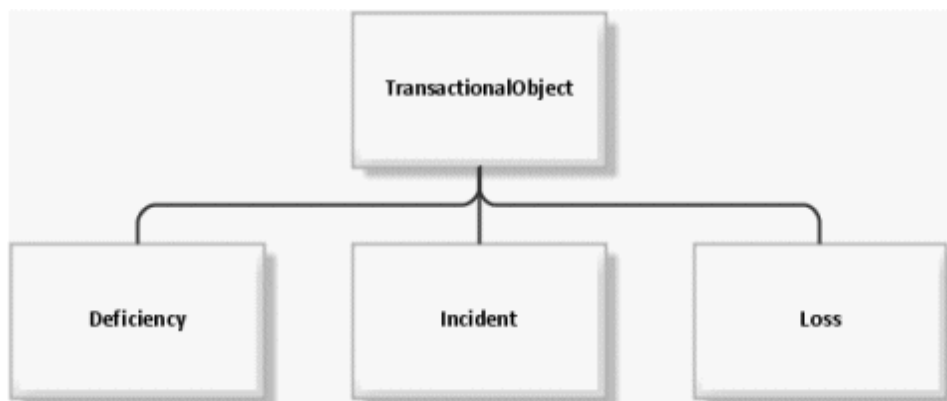
The **transactionalObject** object type (objectTypes.xml: **TRANSACTIONAL**) combines several attributes that belong to the typical transactional data objects. Usually, these object types (e.g., Test case) pass through various roles (e.g., Tester and Reviewer) during a workflow and represent the basis of the data recorded in ARIS Risk & Compliance Manager.

ATTRIBUTES

Attribute ID	Data type	Usage
owner_status	Enumeration	Status
owner_group	Assignment	Group responsible for execution
owner	Assignment	Executing user
owner_substitute	Assignment	Substitute of executing user
execution_date	Date	Execution date
reviewer_status	Enumeration	Status of review
reviewer_group	Assignment	Group responsible for review
reviewer	Assignment	Reviewer
reviewer_substitute	Assignment	Substitute of reviewer
review_date	Date	Review date

If the two status attributes require different enumerations for different workflows, you can overwrite the corresponding attributes of the actual (inheriting) object type. For group assignment attributes, the roles of assignable groups need to be specified at the inheriting object type.

INHERITANCE DIAGRAM



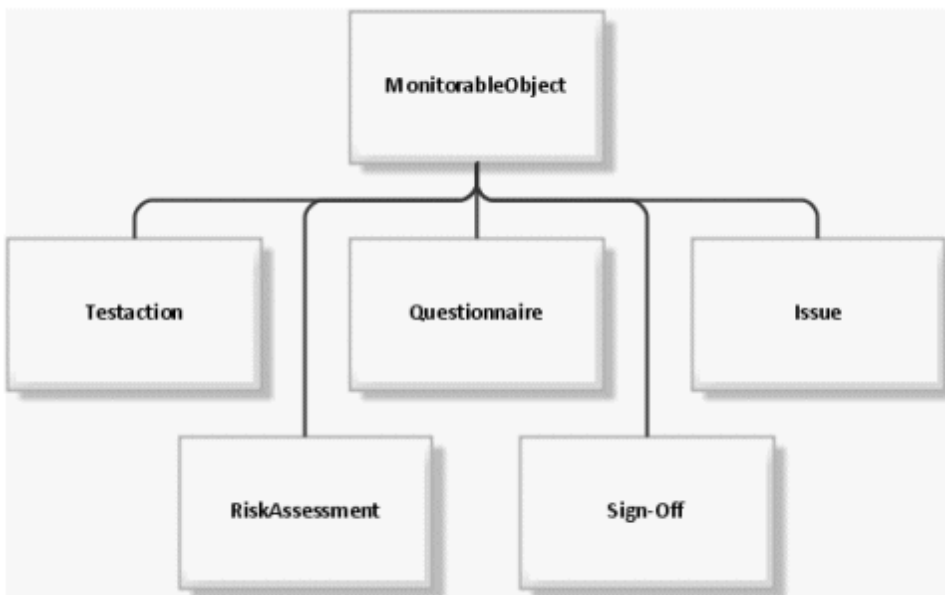
3.4.4 MonitorableObject object type

The **monitorableObject** object type (objectTypes.xml: **MONITORABLE**) adds several attributes to the **transactionalObject** object type described previously, which are associated with time-based monitoring. Transactional data types with expiration dates (e.g., Test case) monitored by the application are supposed to inherit from this object type.

ATTRIBUTES

Attribute ID	Data type	Usage
plannedstartdate	Enumeration	Start date of processing period
plannedenddate	Assignment	End date of processing period
controlstartdate	Assignment	Start date of control period
controlenddate	Assignment	End date of control period

INHERITANCE DIAGRAM



3.4.5 RecurringObject object type

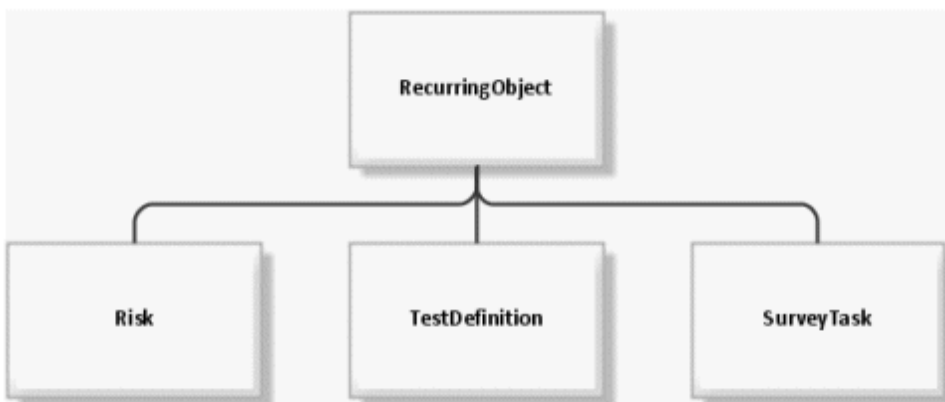
The **RecurringObject** object type (objectTypes.xml: **RECURRING**) is part of the master data. It combines attributes required for the regeneration of transactional data objects.

ATTRIBUTES

Attribute ID	Data type	Usage
owner_group	Assignment	Group responsible for execution
frequency	Enumeration	Frequency used to generate the transactional data objects (once, daily, weekly, etc.)
duration	Integer (long)	Time limit for execution in days
startdate	Date	Date from which transactional data is generated regularly
enddate	Date	Date up to which transactional data is generated regularly
control_period	Enumeration	Length of the control period (day, week, month, etc.)
offset	Integer (long)	Offset of the control period in days
reviewer_group	Assignment	Group responsible for review

If the **frequency** and **control_period** attributes require different enumerations for different workflows, you can overwrite the corresponding attributes of the actual (inheriting) object type. For group assignment attributes, the selection must be restricted to one role at the inheriting object type. For this, you must overwrite this attribute and assign the appropriate restriction to it.

INHERITANCE DIAGRAM



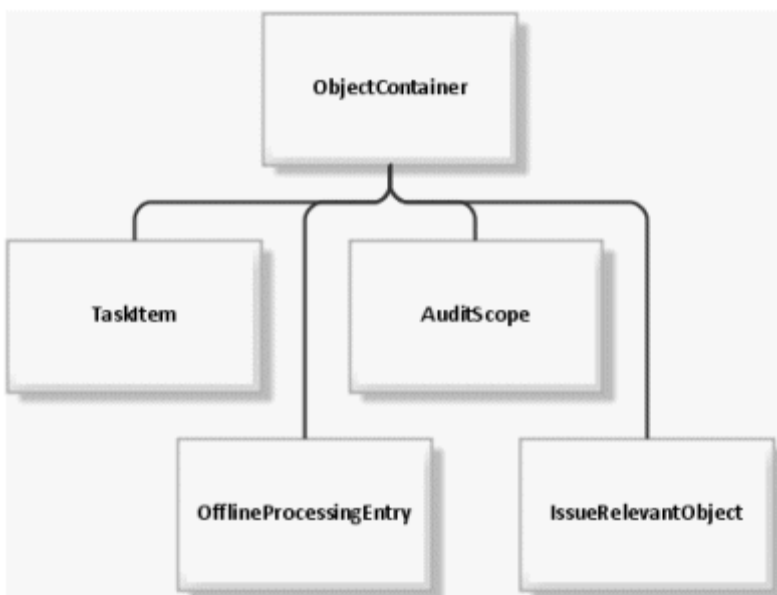
3.4.6 ObjectContainer object type

The **objectContainer** object type (objectTypes.xml: **OBJECTCONTAINER**) serves as a container for other objects. It is used in Issue Management, for example, to connect objects of any type as issue-relevant objects.

ATTRIBUTES

Attribute ID	Data type	Usage
object_id	Integer (long)	ID of the object contained
object_version_number	Integer (long)	Version number of the object contained
object_objtype	String	Object type of the object contained
object_clientSign	String	Auxiliary attribute for the environment filter
object_clientSigns	String	Auxiliary attribute for the environment filter (contains a comma-separated list of assigned environment identifiers)
object_name	String	Name of the object contained
object_ovid	String	Auxiliary attribute for the selection (object version ID)
role	Enumeration	Role used for accessing the object contained

INHERITANCE DIAGRAM



3.4.7 Inheritance in the file `objectTypes.xml`

Inheritance is expressed using the XML attribute **extends** at the XML element **objectType** in the file **objectTypes.xml**. The value of the attribute must contain the ID of the superior object.

BASIC OBJECTS WITH A SPECIFIC MEANING

OBJECT, VERSION, TRANSACTIONAL, RECURRING, MONITORABLE, OBJECTCONTAINER

INHERITANCE STRUCTURE

USERPROFILE > OBJECT

ISSUE->MONITORABLE > TRANSACTIONAL > VERSION > OBJECT

INCIDENT > TRANSACTIONAL > VERSION > OBJECT

JOBINFORMATION > OBJECT

OPTION > VERSION > OBJECT

POLICYREVIEWTASK > RECURRING > VERSION > OBJECT

AUDIT > MONITORABLE > TRANSACTIONAL > VERSION > OBJECT

SUBSCRIPTION > OBJECT

DOCUMENTLINKTYPE > OBJECT

TASKITEM > OBJECTCONTAINER > OBJECT

CHANGEREVIEW > MONITORABLE > TRANSACTIONAL > VERSION > OBJECT

VERSION > OBJECT

OFFLINEPROCESSINGENTRY > OBJECTCONTAINER > OBJECT

HIERARCHY > RECURRING > VERSION > OBJECT

OBJECTCONTAINER > OBJECT

INTERNALMESSAGE > OBJECT

AUDITSTEP > MONITORABLE > TRANSACTIONAL > VERSION > OBJECT

JOBQUEUEENTRY > OBJECT

DEFICIENCY > VERSION > OBJECT

SOPROCESS > MONITORABLE > TRANSACTIONAL > VERSION > OBJECT

RISKASSESSMENT > MONITORABLE > TRANSACTIONAL > VERSION > OBJECT

DOCUMENT > OBJECT

OBJECT > OBJECT

QUESTIONNAIRESECTION > OBJECT

BOOKMARK > OBJECT

SURVEY > MONITORABLE > TRANSACTIONAL > VERSION > OBJECT

AUDITSCOPE > OBJECTCONTAINER > OBJECT

USERGROUP > VERSION > OBJECT

LOSS > TRANSACTIONAL > VERSION > OBJECT

MONITORABLE > TRANSACTIONAL > VERSION > OBJECT
POLICYDEFINITION > RECURRING > VERSION > OBJECT
MESSAGETEMPLATES > OBJECT
RECURRING > VERSION > OBJECT
POLICYREVIEW > MONITORABLE > TRANSACTIONAL > VERSION > OBJECT
EXCEPTION > VERSION > OBJECT
SOTASK > RECURRING > VERSION > OBJECT
OPTIONSET > VERSION > OBJECT
ENVIRONMENT > VERSION > OBJECT
CONTROL > VERSION > OBJECT
SECTION > VERSION > OBJECT
TESTDEFINITION > RECURRING > VERSION > OBJECT
QUESTIONNAIRE_TEMPLATE > VERSION > OBJECT
ISSUERELEVANTOBJECT > OBJECTCONTAINER > OBJECT
SURVEYTASK > RECURRING > VERSION > OBJECT
ANSWER > TRANSACTIONAL > VERSION > OBJECT
SITE > VERSION > OBJECT
AUDITTEMPLATE > RECURRING > VERSION > OBJECT
POLICYCONFIRMATION > MONITORABLE > TRANSACTIONAL > VERSION > OBJECT
TRANSACTIONAL > VERSION > OBJECT
RISK > RECURRING > VERSION > OBJECT
OBJ2OBJ > OBJECT
QUESTION > VERSION > OBJECT
TESTCASE > MONITORABLE > TRANSACTIONAL > VERSION > OBJECT
USER > VERSION > OBJECT
POLICYAPPROVAL > MONITORABLE > TRANSACTIONAL > VERSION > OBJECT
AUDITSTEPTEMPLATE > RECURRING > VERSION > OBJECT
POLICY > MONITORABLE > TRANSACTIONAL > VERSION > OBJECT
RECOMMENDATION > OBJECT
NEWSMESSAGE > VERSION > OBJECT
QUESTIONNAIRE > MONITORABLE > TRANSACTIONAL > VERSION > OBJECT

3.5 Conventions

3.5.1 Conventions in the XML configuration

ARIS Risk & Compliance Manager version 4.0 introduces numerous conventions that reduce the configuration effort. Examples of these conventions are property keys consisting of the object and attribute names, and buttons subject to naming conventions. The keys and file names are determined via the name convention and the corresponding resources are automatically loaded. The function of these conventions is documented in the relevant schema (XSD) for each XML file.

3.5.2 Conventions for object generation

Conventions are used for object generation to reduce the customizing effort. A significant example is the automatic transport of attributes from master data to transactional data. For example, the **test activities** attribute (testingsteps) is automatically transported during test case generation from the test definition to the generated test case via a name convention. When a new attribute to be transported to a transactional data object was introduced in earlier ARIS Risk & Compliance Manager versions, not only the XML configuration, but also the corresponding object generator had to be adapted in the Java source code. From version 4.0, you must assign identical names to the attributes at the source object and target object.

3.5.2.1 Environment association in environment-specific objects

For environment-specific objects, the environment association of the source object is automatically transferred to the target object.

3.5.2.2 MonitorableObject object type

During the generation of objects of the **monitorableObject** object type, attributes are transferred by the relevant **recurringObject** and calculated based on the master data attributes, such as start and end date. This requires an appropriate **recurringObject** to exist in the context. The default ARIS Risk & Compliance Manager configuration includes the following relationships between recurring and monitoring objects:

TESTDEFINITION > TESTCASE

SURVEYTASK > SURVEY

SURVEYTASK > QUESTIONNAIRE

RISK > RISKASSESSMENT

The following table illustrates how the attributes are handled:

Attribute	Handling
plannedstartdate	Is calculated based on the start date (startdate) and the frequency (frequency) of the source object.
plannedenddate	Is calculated based on the start date of the target object (plannedstartdate) and from the duration of the source object (duration).
controlenddate	Is calculated based on the start date of the target object (plannedstartdate) and from the offset of the source object (offset).
controlstartdate	Is calculated based on the end of the control period (controlenddate) and the control period (control_period) of the source object.
owner_group	Is directly transferred from the source object.
reviewer_group	Is directly transferred from the source object.

3.5.2.3 Identical attribute names

Target object attributes having attributes with identical names at one of the source objects are automatically transferred. This does not apply to attributes inherited from one of the basic object types **Object (OBJECT)** or **VersionObject (VERSION)**. It can be necessary to assign identical names to the attributes but to prevent that the values are automatically transferred. To suppress the automatic transfer of values, you can pass a list of attribute names you want to skip to the help class in which the conventions are applied.

3.5.2.4 Object assignment if names are identical

Target objects that are transactional data are often linked with source objects as attributes in order to provide additional information to the end user processing the target object for a better understanding of the task at hand. Source objects are linked automatically from ARIS Risk & Compliance Manager version 4.0, provided the name of the attribute at the target object matches the name of the object type of a source object. For example, when a test case is generated, the **risk** attribute is assigned the corresponding risk (**RISK**) that was transferred as a source object.

3.6 Class mappings

A class mapping links a specific implementation (class) with a name. These names are used in other parts of the ARIS Risk & Compliance Manager configuration to refer to the required implementation. The name is significantly shorter and catchier than the rather long name of the class. This ensures that the configuration remains plausible. Various class mappings are available in ARIS Risk & Compliance Manager. Their definition, usage, and scope of application are described in the following section.

3.6.1 Actions

Action commands are used in the user interface to convert user interaction to business logic commands. In most cases, the default implementation is sufficient. However, in some special cases or in customizing, you must adapt or supplement the behavior. For this, use a class mapping which simplifies customizing. All classes associated with this mapping must contain the **IActionCommand** interface.

The definition of Action command mappings consists of several parts. One part is the definition of **ActionCommandIds**, which contains a list of all valid commands and their explanation.

```
<commandIds>
<commandId id="create"      description="create new objects" />

<commandId id="delete"     description="delete objects, version objects will
                           be deactivated (see 'reactivate')"/>

<commandId id="duplicate"  description="creates a duplicate out of the
                           selected object"/>

<commandId id="edit"       description="open the selected object for
                           editing"/>

<commandId id="reactivate" description="reactivate deactivated objects" />

<commandId id="save"       description="make changes on objects persistent" />

...

</commandIds>
```

The other parts are:

- **objectTypeCommands**
Define commands that control one or more objects. They are usually used with forms.
- **listCommands**
Define commands that control lists, e.g., paging, applying a filter, etc.
- **evaluationCommands:**
Define commands that control evaluations, e.g., expanding the tree structure, applying a filter, etc.
- **jobCommands**
Define commands that take into account the various job properties and execute the jobs accordingly.

- **dialogCommands**

Define commands that control dialogs.

Each of these areas consists of a list of `<commandSet>` elements. Each set requires a name attribute. This attribute specifies what the following command definitions refer to. For `objectTypeCommands`, it refers to the unique identifier of the object type, for `listCommands`, it is the unique identifier of the list, etc.

A special `commandSet` is "common". It includes the default implementation for all object types, lists, etc., so that only special implementations must be specified individually. For example, if multiple lists use the same implementation, you can enter all lists, separated by commas, as names of the `commandSet`. A list can occur in several `commandSets` at the same time.

```
<listCommands>

  <commandSet name="common">
    <actionCommand commandId="applyFilter" clsName="BaseApplyFilterCommand" />
    ...
  </commandSet>

  <commandSet name="riskList,controlList">
    <actionCommand commandId="applyFilter" clsName="SpecialApplyFilterCommand" />
  </commandSet>

  <commandSet name="riskList">
    <actionCommand commandId="resetFilter" clsName="SpecialResetFilterCommand" />
  </commandSet>

</listCommands>
```

3.6.2 Command class mappings

Commands section

Commands are used in the context of workflows to execute the business logic. They are usually very compact and specialized in one task in order to be reusable. You can assign parameters for the respective operation purpose (command chain) to them. There are also several special implementations that were written for a specific purpose and that can not or hardly be reused. Nevertheless, you can use them as templates for your own commands. Note that commands must implement the **ICommand** interface.

3.6.3 Statistics class mappings

This class mapping encompasses all classes with alias names that are used in the statistics. For details on using these classes, see chapter **Add/adapt statistics** (Page 57). For details on implementing additional classes, see the relevant section of the Java documentation about the interface to be implemented.

EVALUATIONACCESSCONTROL SECTION

Access control implementations are required to grant only specific users access to certain statistics. These classes implement the **IEvaluationAccessControl** interface.

STATISTICTREEPROVIDER SECTION

Tree provider implementations are required to generate the hierarchy structures that the statistics are based on (i.e., the tree in the first column).

STATISTICDATAFILTER SECTION

Data filter implementations are used for filtering data for statistics. These classes implement the **IStatisticDataFilter** interface.

STATISTICDATASOURCE SECTION

Data source implementations are used for configuring the data sources of statistics. The default configuration only offers the data sources **view** and **tree**. **view** grants direct access to the ARIS Risk & Compliance Manager database, **tree** allows the usage of the tree provider as the data source. These classes implement the **IStatisticDataSource** interface.

STATISTICCALCULATOR SECTION

Calculator implementations are used for processing the data to be displayed. They convert the technical data provided by data source implementations into data readable by users. These classes implement the **IStatisticCalculator** interface.

STATISTICDATALINKER SECTION

Data linker implementations are used for linking the data to be displayed. For example, you can link to a detailed view of the data in the form of a list or an additional statistic. These classes implement the **IStatisticDataLinking** interface.

3.6.4 BI class mappings

PredefinedValueProvider section

A value provider allows automatic generation of selection options in a dialog based on the user context and additional parameters, if applicable. These selection boxes occur often and their content is similar, but user-defined. Various default implementations cover a lot of cases.

3.6.5 UI class mappings

This section provides the mappings that design the user interface.

RENDERER SECTION

A renderer generates an HTML fragment to represent an attribute in forms or lists. A renderer must implement the **IRenderer** interface. In addition to the class for HTML representation (attribute **reportClsName**), you can specify a class to adapt the representation in PDF/Excel reports (attribute **clsName**).

FILTERRENDERER SECTION

Filter renderers differ from general renderers only with regard to their implementation. Here, some particular cases are considered that concern the representation of filters.

COLUMNRENDERER SECTION

Column renderers implement the **IColumnRenderer** interface and are used by the configurable statistics to display data cells.

LAYOUTER SECTION

Layouters generate HTML fragments by combining one or more renderers suitable for the relevant **control**. Layouters implement the **ILayouter** interface.

CONTROLS SECTION

Controls, such as a form or a list, combine the HTML fragments of the layouters and add further elements, such as buttons in order to generate a completely interactive HTML page. A control can consist of several components. A control can refer to another control (attribute **extends**), by overwriting parts of a control or its components. In this case, the components to be changed must be redefined.

```
<control name="statistic"          clsName="Statistic" >
  <component name="footer"        clsName=" StatisticFooter" />
  <component name="header"        clsName=" StatisticHeader" />
  <component name="row"           clsName=" StatisticDataRow" />
  <component name="toolbar"       clsName=" StatisticToolbar" />
  <component name="treeNode"     clsName=" StatisticTreeNode" />
</control>
<control name="scoping" extends="statistic" clsName="Statistic" >
  <component name="toolbar"      clsName="ScopingStatisticToolbar" />
</control>
```

3.6.6 View class mappings

In order to be able to customize the data of a configured view, you can specify an additional handler class at a **<view>** element using the **viewHandler** attribute. This handler class must implement the **IViewHandler** interface. Then, it is possible within this class to customize the data returned using additional details. This form of customizing is necessary if the additionally required adaptations cannot be configured completely.

3.6.7 VCREG.XML configuration file

The configuration file **vcreg.xml** registers the validator and the converter to be used for the attributes of ARIS Risk & Compliance Manager object definitions (see **Assign validator** (Page 20)). In doing so, a name that refers to a fully qualified class name is defined for the validator or for the converter.

Example

```
<validator name="minlength"
clsName="com.idsscheer.webapps.arcm.bl.models.objectmodel.attribute.vc.validator.MinLengthValidator"
propertyKey="errors.minlength"/>
```

In this example, validator **minlength** is defined with its implementation being listed in the **clsName** attribute. For validators, a property is listed in the **propertyKey** attribute, which is displayed in the user interface if the validation is negative.

3.7 Customize help

The help can be extended by creating new HTML pages or customized by creating links to existing pages. In addition, the content of existing pages can be adapted.

Location	Property file in the properties/help folder
Procedure	<ol style="list-style-type: none"> 1. Copy the HTML page to be used for the help to webapps\arcm\help\<language code>\embedded. The name of the file must be the same as the corresponding help ID. 2. After the page is copied to the Help folder enter the new property key in a new line in the property file stated above. 3. Assign the new help ID as a value to this new property key. <p>A common properties file is available for all languages.</p>
Remark	<p>According to the convention, a help page is automatically expected when a new form, list, or statistics is created. If a help page is not needed at this point, enter the property key defined in the convention without a value assigned in the properties file. In this case, a help button is not displayed.</p> <p>If you do not want to create a new help page, you can refer to an existing help page. You can create this link as described above by assigning the existing help ID in one of the appropriate property files.</p> <p>Conventions for naming new property keys:</p> <p>Form: ARCM_FORM_[FORM ID].PAGE_[PAGE ID].HLP</p> <p>List: ARCM_LIST_[LIST ID].HLP</p> <p>Statistics: ARCM_EVALUATION_[EVALUATION ID].HLP</p>

4 Basic use cases

4.1 Customize object properties

4.1.1 Overwrite the schema version

As soon as you customize object properties, you must overwrite the schema tag from the file **objectTypes.xml** in customizing. This is the only way to ensure that data exports and imports from the customized ARIS Risk & Compliance Manager version can be properly assigned to the relevant version. The customizing of the version also serves as a fixed starting point for future migrations. If you do not overwrite the schema tag, but still make changes to the schema, it will not be possible to start the ARIS Risk & Compliance Manager Server.

Procedure

Enter the name of the customer project without blanks in the customizing attribute in the overwritten schema tag. This attribute has the value **standard** if the schema has not been customized.

Example

Entry for a customer project called **United Motor Group** on the basis of ARIS Risk & Compliance Manager version 4.0.0.2:

```
<schema version="arcm_4.0.0.2" customizing="UnitedMotorGroup" />
```

If different customer versions based on a single ARIS Risk & Compliance Manager version are to be supplied, you can indicate this by stating a version number in the project name.

Example

Entry for a customer project called **United Motor Group version 1** on the basis of ARIS Risk & Compliance Manager version 4.0.0.2:

```
<schema version="arcm_4.0.0.2" customizing="UnitedMotorGroup_v1" />
```

If changes in the ARIS Risk & Compliance Manager schema are due to a change in the data import from ARIS Architect, you must also adapt the target schema in the relevant mapping file **Aris2arcm-mapping_[APPROACH].xml**.

Procedure

1. Find the infoHeader tag with the **standard** attribute:
 `schema_version="arcm_4.0.0.2_rba_standard"`
2. Replace "standard" with the name of the customer project.

Example

```
schema_version="arcm_4.0.0.2_rba_UnitedMotorGroup "
```


4.1.2 Add/adapt a simple attribute

4.1.2.1 Create a simple attribute

4.1.2.1.1 Adapt an object type

To adapt an object type, copy the original to the customizing file. Then you can change the properties and attributes of the object type.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy the <objectType> element from the default configuration to the customizing file. 2. Create new attributes within the <objectType> element. You must set at least the id property. The value must be unique within the object type.
Documents	objectTypes.xml, objectTypes.xsd
Example	ModifyObject_AddSimpleAttribute \WEB-INF\config\custom\xml\custom.xml: Add Simple Attribute

4.1.2.1.2 Add/adapt properties

Properties are used for the multi-language capability of the application. A separate file is available for each language. These files include the country code as a suffix in their names.

Location	Property file in the properties/application folder
Procedure	Enter the new property key followed by an equal sign and the corresponding translation in a new line. A separate file is available for each language.
Documents	See Customize names.
Example	ModifyObject_AddSimpleAttribute \WEB-INF\config\custom\properties\application\custom.properties: Add Simple Attribute

4.1.2.1.3 Assign validator

To ensure that only specific or permitted values are included in the database you can assign a validator to an attribute.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none">1. Add a new <validate> element as a subordinate element of the attribute.2. Specify a registered validator as a name.
Documents	vcreg.xml, vcreg.xsd
Example	ModifyObject_AddSimpleAttribute \\WEB-INF\\config\\custom\\xml\\custom.xml: Add Simple Attribute

4.1.2.1.4 Assign converter

It can be necessary to assign a converter to the attribute that modifies the data between application and database. Especially the **startdate** and **enddate** converters are often used if two date fields are to describe a time period.

Location	XML file in the xml folder
Procedure	Add a new <convert> element as a subordinate element of the attribute and specify a registered converter as a name.
Documents	vcreg.xml, vcreg.xsd
Example	See Assign validator (Page 20).

4.1.2.2 Add an attribute to a form

4.1.2.2.1 Adapt a form

For attributes to be displayed and for you to be able to edit them in the user interface, they must be specified in the object type form. As the order of the attributes can be important, it is defined separately from the object type definition.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy the <form> element from the default configuration to customizing file, and add a new <row> element at the position where the new attribute is to be displayed. 2. Create a subnode <element> with the property attribute.idref. You must specify a unique name for the attribute.
Documents	forms_[module].xml, forms.xsd
Example	ModifyObject_AddSimpleAttribute \WEB-INF\config\custom\xml\custom.xml: Modify Form

4.1.2.2.2 Add/adapt properties of a form

See **Add/adapt properties** (Page 32).

Procedure	Define the property PropertyKey of the <row> element.
Remark	With forms, this is an optional step and only necessary if the property key of the first subnode <element> that is used according to the convention does not provide an appropriate description. If, for example, there is a start date and an end date in one row, you should define a new property key reflecting the name of the relevant period.
Example	ModifyObject_AddSimpleAttribute \WEB-INF\config\custom\properties\application\custom.properties: Modify Form

4.1.2.2.3 Assign a renderer

A default representation is defined for all attributes. To change the default representation, you must specify a different class generating this representation, or you adapt the default representation by means of a default renderer using parameters.

Location	XML file in the xml folder
Procedure	Assign a valid and registered renderer to the template property of <element> .
Documents	uiClassMappings.xml, uiClassMappings.xsd
Example	ModifyObject_AddSimpleAttribute \WEB-INF\config\custom\xml\custom.xml: Assign Renderer

4.1.2.2.4 Adapt rules

New attributes are optional, visible, and not editable by default. To change this, you must adapt the rules. Usually, there are already rules that determine whether the attributes are visible and editable. If the conditions of the rule are met, the new attribute can simply be added. If other conditions are required, you must create a new rule.

Location	DRL file in the rules folder.
Procedure	<ol style="list-style-type: none"> 1. Copy the default file to customizing. 2. Modify rules or add new ones.
Documents	See chapter Add/adapt rule (Page 71).
Example	ModifyObject_AddSimpleAttribute \WEB-INF\config\custom\rules\usergroup.drl: Modify Form

4.1.2.2.5 Add/adapt reports

Normally, reports are automatically generated based on the form definitions. However, you can modify and configure the reports.

Location	XML file in the xml folder
Documents	Chapter Add/adapt reports (Page 63)
Example	ModifyObject_AddSimpleAttribute \WEB-INF\config\custom\xml\custom.xml: Modify Report

4.1.2.3 Add an attribute to a list

4.1.2.3.1 Adapt a list

The modification of lists is similar to the modification of forms.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Add a new <column> element at the relevant position. 2. Specify a name unique in the list for the id property. The attribute.idref property must refer to a valid data retrieval column. 3. Add a new <listHeader> element. To enable sorting, the column property must point to the corresponding column name. 4. Complete the property key and the width property. The value for width is usually specified in relative widths and its sum should not exceed 98%. The remaining 2% are occupied by the buttons at the beginning and end of each row.
Documents	lists_[module].xml, lists.xsd
Example	ModifyObject_AddSimpleAttribute \WEB-INF\config\custom\xml\custom.xml: Modify List

4.1.2.3.2 Add/adapt properties of a list

See **Add/adapt properties** (Page 19).

Example

ModifyObject_AddSimpleAttribute

\WEB-INF\config\custom\properties\application\custom.properties: Modify List

4.1.2.3.3 Adapt data retrieval for a list

In the list, a view defined is defined as a data source. This view must be customized for the new attribute to be included.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Insert a new <viewColumn> element as a subordinate element in the <viewObject> element. The view object defines the object type of the view columns. 2. Assign a unique name to the id property. The attributeType property references an attribute of the associated object type. If you do not only want to filter, but also output the column, you must set the isSelectColumn property to true.
Documents	view_[module].xml, views.xsd
Example	ModifaObject_AddSimpleAttribute \WEB-INF\config\custom\xml\custom.xml: Modify View

4.1.2.3.4 Add a renderer

A default representation is defined for all attributes. To change the default representation, you must specify a different class generating this representation, or you adapt the default representation by means of a default renderer using parameters.

Location	XML file in the xml folder
Procedure	Assign a valid and registered renderer to the template property of <column> .
Documents	uiClassMappings.xml, uiClassMappings.xsd
Example	ModifyObject_AddSimpleAttribute \WEB-INF\config\custom\xml\custom.xml: Assign List Renderer

4.1.2.3.5 Add/adapt reports

See **Add/adapt reports** (Page 63).

4.1.2.4 Add an attribute to a filter

4.1.2.4.1 Adapt a list filter

All data retrieval columns can be filtered, regardless of whether they exist in the result or not. You can modify and save the filter values for reuse.

Two filter views exist. One is the view as a simple list in the **Save/configure filter** dialog, which is similar to an object form, and another is the view as a drop-down filter directly above the list. The format for the **Save/configure filter** dialog is specified during filter definition. The filter definition is similar to the form definition and works according to the same principle.

You can customize the order and arrangement of the various filterable attributes for the drop-down filter. All filter attributes that are not explicitly arranged are added at the end of the extended part of the filter in line with the order of the filter definition.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy the relevant filter to customizing. 2. Insert a new <filterRow> element at the relevant position of the filter definition. 3. Below the new element, create a subordinate element <filterElement> and set the dataReference.idref property to the relevant viewColumn. The template and filterType properties are determined automatically, but they can also be set explicitly if another representation (template) or data storage (filterType) is desired or required.
Documents	filters_[module].xml, uiClassMapping.xml, biClassMapping.xml, filter.xsd, uiClassMapping.xsd, biClassMapping.xsd
Example	ModifyObject_AddSimpleAttribute \WEB-INF\config\custom\xml\custom.xml: Modify filter

4.1.2.4.2 Add/adapt properties of a filter

See **Add/adapt properties** (Page 19).

Example

ModifyObject_AddSimpleAttribute

\WEB-INF\config\custom\properties\application\custom.properties: Modify filter

4.1.2.4.3 Assign a renderer

See [Assign a renderer \(form\)](#) (Page 22).

4.1.3 Add/modify an enumeration attribute

4.1.3.1 Create an enumeration attribute

4.1.3.1.1 Add/adapt an enumeration

For a new enumeration attribute, you can reuse an existing enumeration or create a new one. It is recommended that you create a new enumeration. Reusing an existing enumeration when customizing elements can lead to conflicts with other enumeration attributes that refer to this enumeration.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Insert a new <enum> element. 2. Define the required properties id (unique name), isMultiple (single or multiple selection), and type (number or string) 3. As subordinate elements, add the elements of the enumeration as an <enumitem> element. 4. Assign a unique name (id) and a corresponding value (value) to each <enumitem> element.
Remark	If number type is selected, the value property of the <enumitem> elements must only contain figures. If the formRelevant property is set to false , you cannot select the value in the form. The same applies to filterRelevant with regard to the selectability in the filter.
Documents	enumerations.xml
Example	ModifyObject_AddEnumerationAttribute \WEB-INF\config\custom\xml\custom.xml: Add/Modify enumeration

4.1.3.1.2 Add/adapt properties of an enumeration

See [Add/adapt properties](#) (Page 19).

Remark	According to the convention, the property key of an enumeration element looks as follows: enumeration.[Name of enumeration].[Name of element].DBI
Example	ModifyObject_AddEnumerationAttribute \WEB-INF\config\custom\properties\application\custom.properties: Add/Modify enumeration

4.1.3.1.3 Adapt an object type

See **Adapt an object type** (Page 19).

Procedure	In addition to the simple attributes, you must specify the enumeration property. This property specifies the enumeration to be used for the attribute.
Example	ModifyObject_AddEnumerationAttribute \\WEB-INF\config\custom\xml\custom.xml: Add enumeration attribute

4.1.3.1.4 Add an attribute to a form

See **Add an attribute to a form** (Page 21).

Example

ModifyObject_AddEnumerationAttribute

\\WEB-INF\config\custom\xml\custom.xml: Modify form

ModifyObject_AddEnumerationAttribute

\\WEB-INF\config\custom\rules\usergroup.drl: Modify Form

4.1.3.1.5 Adapt data retrieval for a list

See **Adapt data retrieval for a list** (Page 24).

Example

ModifyObject_AddEnumerationAttribute

\\WEB-INF\config\custom\xml\custom.xml: Modify view

4.1.3.1.6 Add an attribute to a list

See **Add an attribute to a list** (Page 23).

Example

ModifyObject_AddEnumerationAttribute

\\WEB-INF\config\custom\xml\custom.xml: Modify list

4.1.3.1.7 Add an attribute to a filter

See **Add an attribute to a filter** (Page 25).

Example

ModifyObject_AddEnumerationAttribute

\\WEB-INF\config\custom\xml\custom.xml: Modify filter

4.1.4 Add/adapt a list attribute

4.1.4.1 Create a list attribute

4.1.4.1.1 Adapt an object type

See **Adapt an object type** (Page 19).

Remark	List attributes require additional properties: <ul style="list-style-type: none">▪ maxSize indicates the maximum number of objects that can be assigned to the attribute. -1 means an infinite number of objects.▪ The linkType property requires a unique numeric value.▪ objectType.idref contains the comma-separated names of all object types that can be assigned. Generally, this should be only one object type.▪ The orderType property indicates how assigned objects are to be sorted.
Documents	objectTypes.xsd
Example	ModifyObject_AddListAttribute \\WEB-INF\\config\\custom\\xml\\custom.xml: Add list attribute

4.1.4.1.2 Add/adapt properties

See **Add/adapt properties** (Page 19).

Example

ModifyObject_AddListAttribute

\\WEB-INF\\config\\custom\\properties\\application\\custom.properties: Add list attribute

4.1.4.1.3 Adapt list restrictions

In addition to the object type restriction, there is another option to further restrict the number of objects allowed. This restriction applies to objects whose attributes have specific values.

Location	XML file in the xml folder
Procedure	<p>Add a <listRestriction> element as a new subordinate element. It has no property, but merely serves to group the <attributeRestriction> subordinate elements. They have the following two mandatory properties:</p> <ul style="list-style-type: none">▪ attribute refers to an attribute of an object type that is permitted for list attributes.▪ value refers to a value that is permitted for the attribute. <p>Objects with other than the permitted values cannot be assigned.</p>
Remark	<p>You should only refer to attributes that no longer change within the life cycle of the object, such as the type of a hierarchy object. <listRestriction> elements of a list attribute are OR-connected. <listRestriction> elements of a list restriction are AND-connected.</p>
Documents	objectTypes.xsd
Example	<p>ModifyObject_AddListAttribute</p> <p>\WEB-INF\config\custom\xml\custom.xml: Add list restrictions</p>

4.1.4.1.4 Adapt roles

For list attributes, privileges must be assigned explicitly. These privileges specifies which roles are permitted to modify list attributes. You can individually set the privilege for adding objects to and for removing objects from a list attribute. By default, no role has the privilege to do this. You must explicitly assign this privilege to all roles.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Add a new <roles> element. 2. Add one or more <role> subordinate elements that you can copy from the default configuration. 3. Customize the privileges. The roles need privileges for adding objects to or for removing them from the new list attribute. 4. Add a new <relation> subordinate element to the <object> element references the matching object type. The right.idref property knows the possible values a (attach), r (remove), and ar (attach/remove). listAttrType.id references the corresponding list attribute.
Documents	roles.xml, roles.xsd
Example	ModifyObject_AddListAttribute \WEB-INF\config\custom\xml\custom.xml: Modify roles

4.1.4.1.5 Add an attribute to a form

See **Add an attribute to a form** (Page 21).

Procedure	<p>There are additional properties you must specify for list attributes. If the list attribute can be edited by the user, a selection list that displays the admissible objects is required.</p> <p>Two options are available. The selection list can be generated automatically or you can manually define a list (see Add a selection list (Page 32)). The second option is relevant if the selection list contains special cases that the automatic lists cannot cover.</p> <p>You can configure the selection list using a <parameter> subordinate element with the name selectionList and specifying the name of the selection list as value. For an automatically generated selection list, the value for value must be AUTO.</p> <p>You can also customize the buttons for managing and manipulating the objects. Open, Edit, Create assignment and Remove assignment are the default buttons. They have the subordinate elements <button.add> and <button.remove>. The button.idref property defines the button to be added or removed. location defines whether the button is to be inserted into the toolbar (toolbar) or displayed individually for each assigned object (row). type defines whether the button is only available if the user can edit the list attribute (writable) or if it is always available (always).</p>
Remark	<p>An automatic selection list includes the environment of the object, the list restriction of the list attribute, and the objects already assigned. The primary column of the selection list contains the attribute specified in the nameAttribute property of the selection object type. The other columns and filters result from the attributes defined in the descriptionAttributes property.</p>
Documents	objectTypes.xsd
Example	<p>ModifyObject_AddListAttribute</p> <p>\WEB-INF\config\custom\xml\custom.xml: Modify form</p>

4.1.4.2 Add a selection list

4.1.4.2.1 Adapt a selection list

Adding a selection list is similar to creating a normal list. It includes data retrieval, a representation of the list, and a filter definition. To create a selection list from this normal list, you must configure a few properties.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Create the list and specify the following properties of the <list> element: <ol style="list-style-type: none"> a. listType receives the value selection. b. destDataType.idref receives as a value the object type of the list attribute. c. destAttributeType.idref receives the name of the list attribute itself.
Documents	lists.xsd
Example	ModifyObject_AddListAttribute \WEB-INF\config\custom\xml\custom.xml: Add selection list

4.1.4.2.2 Add/adapt properties

See **Add/adapt properties** (Page 19).

Remark	Omitting the right part in the property file results in no help button being displayed for this element.
Example	ModifyObject_AddListAttribute \WEB-INF\config\custom\properties\help\custom_helpids.properties: Add selection list

4.1.4.2.3 Adapt data query for selection list

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Supplement the data query by two <viewCondition> elements in addition to the other conditions. 2. Set a condition for the obj_id attribute of the main object type with the compare type NOTIN. currentObjectType.id and currentAttributeType.id again refer to the list attribute and its object types. The second condition filters the client_sign attribute. currentObjectType.id receives the same value as before, currentAttributeType.id receives the value client_sign.
Documents	views.xsd
Example	ModifyObject_AddListAttribute \WEB-INF\config\custom\xml\custom.xml: Add selection view

4.1.4.2.4 Assign a renderer

See **Assign a renderer** (Page 22).

4.1.4.2.5 Add a selection list filter

See **Add an attribute to a filter** (Page 25).

Remark	This is the same as adding an attribute a normal filter.
Documents	filters.xsd
Example	ModifyObject_AddListAttribute \WEB-INF\config\custom\xml\custom.xml: Add selection filter

4.2 Customize the object life cycle

The object life cycle is mainly controlled or configured by the following configuration files:

- Workflow configuration: XML files (workflow_*.xml) in the **xml** folder
- Command chain catalog configuration: XML files (commandChains_*.xml) in the **xml** folder
- Rule configuration: DRL files in the **rules** folder (see chapter **Add/adapt rule** (Page 71))
- Generator: Scheduled generation of objects by executing the **<prepare>** and **<insert>** transitions
- Monitor: Scheduled check of processing periods and change of attribute values.

Workflow configuration, rules, and command chains are related as follows:

- The rules define the attribute states (e.g., "can be changed", "visible") that apply to an object in a particular state. Use the state ID in the rules for this. For details, see **Add/adapt rule** (Page 71).
- The command chains contain the commands that are carried out when a transition is executed.

The following chapters explain the configuration of workflows, command chains, and rules.

4.2.1 Workflow configuration

A workflow consists of states and transitions. A state represents the status of an object. This state is defined by the attribute values of an object. Each state must be accessible via at least one transition.

A **transition** represents a transition to another state. A command chain must be defined for each transition. This command chain is executed when the transition takes place. A command chain contains any number of commands. Empty chains are also allowed.

When adjusting a workflow you must specify the new and the changed states. You can omit all of the unchanged states.

The following chapters describe the adding of states and transitions. For details on the configuration of command chains, see **Configuration of the command chain catalog** (Page 42).

4.2.1.1 Add a state

A workflow can consist of up to four different state types. The two following state types can only exist once in a workflow:

- A state that represents a new, unedited and unsaved object (**<state.initial>**).
- A state that represents an unsaved object that was edited by a user or by the system (**<state.prepared>**).

These types can occur multiple times in a workflow, but they must have unique IDs:

- A state that represents an active object (**<state>**), or
- a state that represents a deleted object (**<state.deleted>**).

4.2.1.1.1 Add a state to an active object

An object is active if it is persistent and not deleted, i.e., a user can edit it in a form, for example.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy a <workflow> element from the default configuration to the customizing file. 2. Create one or more new <states> within the <workflow> element. The id attribute must have a value that is unique within the workflow.
Remark	The state of an object must be uniquely identifiable by the values of the <Attribute> elements, i.e., an object is in a particular state if values are defined for it in the <Attribute> elements. The persistent values of an object are used to determine the state. An added state must be accessible by at least one transition. See Add a transition (Page 36).
Documents	workflow_*.xml, workflow.xsd
Example	ModifyObjectLifecycle \WEB-INF\config\custom\xml\testcase_workflow_custom.xml: New custom active workflow state

4.2.1.1.2 Add a state to a deleted object

Users can delete or disable an object by clicking **Delete**. If this object is a versioned object, it is removed from the database, but only deactivated. A deactivated object can be reactivated. For details, see **Add a transition** (Page 36) and **Add a recover transition** (Page 41).

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy a <workflow> element from the default configuration to the customizing file. 2. Create one or more new <state.deleted> within the <workflow> element below the active states. The id attribute must have a value that is unique within the workflow for deleted states.
Remark	The state of an object must be uniquely identifiable by the values of the <Attribute> elements, i.e., an object is in a particular state if it has the values defined in the <Attribute> elements. The persistent values of an object are used to determine the state. An added state.deleted must be accessible by at least one delete transition. See Add a delete transition (Page 40).
Documents	workflow_*.xml, workflow.xsd
Example	ModifyObjectLifecycle \WEB-INF\config\custom\xml\testcase_workflow_custom.xml: New custom deleted workflow state

4.2.1.2 Add a transition

Depending on the state of an object various transitions can be executed. Possible transitions for the active state (<state>) include: **update**, **reset** and **delete**. For deactivated states, only a **recover** transition can be executed to exit the state or to return to an active state. The states **initial.state** and **prepare.state** with the two transitions **prepare** and **insert** are available in order to execute various transitions during the creation of an object.

4.2.1.2.1 Add a prepare transition

A prepare transition is executed when an object is created by a user or a job. After this transition was executed the object is always in the state.prepared state.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy a <workflow> element from the default configuration to the customizing file. 2. Create a new prepare transition within the <state.initial> element. A prepare element must refer to an existing command chain of the command catalog that belongs to the workflow (commandChains_*.xml) in the chain.id attribute.
Remark	<p>Prepare transitions can either have a <permission.workflow> or a <permission.job> subordinate element.</p> <ul style="list-style-type: none"> ▪ <permission.job> is used to execute a transition by the specified job only. ▪ <permission.workflow> is used to execute a transition by the specified workflow only.
Documents	workflow.xsd, workflow_*.xml, commandChains_*.xml
Example	<p>ModifyObjectLifecycle</p> <p>\WEB-INF\config\custom\xml\user_workflow_custom.xml: New custom prepare transition</p>

4.2.1.2.2 Add an insert transition

An insert transition with the **state.prepared** status is executed when an object is saved for the first time. After the transition was executed the object must be in a state defined in the workflow.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy a <workflow> element from the default configuration to the customizing file. 2. Create a new insert transition within the <state.prepared> element. An insert element must always refer to an active state existing in the workflow in the to.state.id attribute, and to an existing command chain of the command catalog (commandChains_*.xml) associated with the workflow in the chain.id attribute.
Remark	<p>Insert transitions can either have a <permission.workflow> or a <permission.job> subordinate element.</p> <ul style="list-style-type: none"> ▪ <permission.job> is used to execute a transition by the specified job only. ▪ <permission.workflow> is used to execute a transition by the specified workflow only.
Documents	workflow.xsd, workflow_*.xml, commandChains_*.xml
Example	<p>ModifyObjectLifecycle</p> <p>\WEB-INF\config\custom\xml\user_workflow_custom.xml: New custom insert transition</p>

4.2.1.2.3 Add an update transition

An update transition is executed when a user or internal process (e.g., job) saves an object in a form. An object can only be saved if it has a state containing an update transition. The Save function is not enabled for objects in a state without outgoing update transition.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy a <workflow> element from the default configuration to the customizing file. 2. Create one or more new update transitions within the <transitions> element. An update element must refer to an active state existing in the workflow in the to.state.id attribute, and to an existing command chain of the command catalog (commandChains_*.xml) associated with the workflow in the chain.id attribute.
Remark	An update transition added must refer to an existing state (to.state.id) and a command chain (chain.id).
Documents	workflow.xsd, workflow_*.xml, commandChains_*.xml
Example	ModifyObjectLifecycle \WEB-INF\config\custom\xml\testcase_workflow_custom.xml: New custom update transition

4.2.1.2.4 Add a reset transition

A reset transition is executed when a user clicks **Reset** in a form, or if an internal process resets the object. An object can only be reset if it has a state containing a reset transition. The Reset function is not enabled for objects in a state without outgoing update transition.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy a <workflow> element from the default configuration to the customizing file. 2. Create one or more new reset transitions within the <transitions> element. A reset element must always refer to an active state existing in the workflow in the to.state.id attribute, and to an existing command chain of the command catalog (commandChains_*.xml) associated with the workflow in the chain.id attribute.
Remark	A reset transition added must refer to an existing state (to.state.id) and a command chain (chain.id).
Documents	workflow.xsd, workflow_*.xml, commandChains_*.xml
Example	ModifyObjectLifecycle \WEB-INF\config\custom\xml\testcase_workflow_custom.xml: New custom reset transition

4.2.1.2.5 Add a delete transition

A delete transition is executed when a user clicks **Delete** in a form, or if an internal process deletes an object. An object can only be deleted if it has a state containing a delete transition. The Delete function is not available for objects in a state without outgoing delete transition.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy a <workflow> element from the default configuration to the customizing file. 2. Create one or more new delete transitions within the <transitions> element. A delete element must always refer to an active state existing in the workflow in the to.state.id attribute, and to an existing command chain of the command catalog (commandChains_*.xml) associated with the workflow in the chain.id attribute.
Remark	A delete transition added must refer to an existing state (to.state.id) and a command chain (chain.id).
Documents	workflow.xsd, workflow_*.xml, commandChains_*.xml
Example	ModifyObjectLifecycle \WEB-INF\config\custom\xml\testcase_workflow_custom.xml: New custom delete transition

4.2.1.2.6 Add a recover transition

A recover transition is executed when a user clicks **Reactivate** in a form of a deleted or deactivated object, or if an internal process reactivates the object. An object can only be reactivated if it has an inactive state containing a recover transition. Other states cannot have outgoing recover transitions. The Reactivate function is not available for inactive objects without outgoing recover transition.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy a <workflow> element from the default configuration to the customizing file. 2. Create one or more delete transitions within the <transitions> element. A recover element must refer to an inactive state existing in the workflow in the to.state.id attribute, and to an existing command chain of the command catalog (commandChains_*.xml) associated with the workflow in the chain.id attribute.
Remark	A recover transition added must refer to an existing active state (to.state.id) and a command chain (chain.id).
Documents	workflow.xsd, workflow_*.xml, commandChains_*.xml
Example	ModifyObjectLifecycle \WEB-INF\config\custom\xml\testcase_workflow_custom.xml: New custom recover transition

4.2.2 Configure the command chain catalog

The command chains (hereinafter referred to as "chains") that can be executed by a specific workflow are grouped in a command chain catalog (hereinafter referred to as "catalog"). The catalog associated with the workflow has the same ID as the workflow itself. You can add new chains to a catalog. A chain does not need to be used by a transition. Deleting unused chains is not necessary. The contents of a chain can be modified, i.e., you can add or remove commands. When adjusting a catalog you must specify the new and the changed chains. You can omit all of the unchanged chains.

4.2.2.1 Modify a command chain

If you want to execute additional actions such as sending messages, changing data, etc. when a transition is run, you must add commands to an existing chain.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy a <catalog> element from the default configuration to the customizing file. 2. Adapt one or more command chains within the <catalog> element by extending the chain with a command element or removing a command element. Each command element has an ID representing a command implementation. These IDs are located in the file commandClassMapping.xml. Detailed information on the use and parameterization of the commands is provided in the Java doc of the relevant command.
Remark	<p>It is possible that the target state of a transition is no longer valid from the perspective of the workflow configuration after a chain was changed.</p> <p>Example</p> <p>If a transition ends in a state that is defined by the value X of the attribute A, but a command sets the value of attribute A to Y, the target state is invalid or not achieved. In this case, the transition execution is undone and an error message is displayed.</p>
Documents	commandChains.xsd, commandChains_*.xml
Example	<p>ModifyObjectLifecycle</p> <p>\WEB-INF\config\custom\xml\testcase_catalog_custom.xml: Change command chain</p>

4.2.2.2 Add a command chain

If a workflow is extended by a new transition and no chain to be used exists, you must add a new chain to the catalog.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy a <catalog> element from the default configuration to the customizing file. 2. Adapt one or more command chains within the <catalog> element. 3. Add any number of command elements to the new chain. A chain can also be empty, i.e., it does not contain a command element. Each command element has an ID representing a command implementation. These IDs are located in the file commandClassMapping.xml. Detailed information on the use and parameterization of the commands is provided in the Java doc of the relevant command.
Remark	<p>After a chain was modified, the target state of a transition may no longer be valid from the perspective of the workflow configuration.</p> <p>Example</p> <p>If a transition ends in a state that is defined by the value X of the attribute A, but a command sets the value of attribute A to Y, the target state is invalid or not achieved. In this case, the transition execution is undone and an error message is displayed.</p>
Documents	commandChains.xsd, commandChains_*.xml
Example	<p>ModifyObjectLifecycle</p> <p>\WEB-INF\config\custom\xml\testcase_catalog_custom.xml: Add custom command chain</p>

4.2.3 Adapt/add user interactions

To work off a command chain it is sometimes necessary to collect additional information from the user. To do so you can configure your own dialogs and have them called up by a command from the chain. The execution of the chain is interrupted at this point and an input mask is displayed. If the mask is filled in correctly the command chain starts again. This time however, it is extended by the input from the user. Generally, there are currently two types of dialogs. The simple dialogs only require a simple confirmation or decision, the more complex dialogs have one or more input boxes.

4.2.3.1 Confirmation dialogs

There are two different confirmation dialogs. The simplest dialog has the **okCancel** ID which, in addition to a question or a note, provides the two buttons **Ok** and **Cancel**. **Cancel** stops the execution of the complete command chain. With **Ok** the execution of the chain continues. The dialog with the **yesNoCancel** ID provides an additional button. In addition to the familiar **Cancel** it also provides the possibility to use **Yes** and **No** which both allow the execution of the command chain to be continued, but allow you to go two different ways in the subsequent commands of the chain, for example, in one case you can send a message and in the other you cannot.

In general, you can use several dialogs in a command chain however, all of the dialogs in a chain must have unique IDs. Information about how new dialogs are defined, for example confirmation dialogs, in the event that you need several of them in a chain, can be found in the following.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Extend the relevant command chain with an additional <code><command></code> element at the relevant point. 2. Use the requestDialog ID. 3. Add two subordinate <code><parameter></code> elements. 4. The first element has the name attribute with the value dialogID and the value attribute with one of the two values okcancel or yesnocancel. 5. The second element has the name attribute with the value propertyKey and the value attribute receives the property key that represents the question asked by the user as a value.
Remark	If you defined your own dialog use its ID.
Documents	commandChains.xsd, commandChains_*.xml
Example	ModifyObjectLifecycle \WEB-INF\config\custom\xml\testcase_catalog_custom.xml: Add custom dialog

4.2.3.2 Input dialogs

If more than one simple confirmation is required, input dialogs are used. This allows you to provide the user with one or more input boxes to fill out.

An attribute is defined for each input box. Similar to the object definitions (**objectTypes.xml**), there are several attribute types available for the individual requirements of the input boxes. You can add validators (**<validator>**) to these attributes, e.g., to limit the scope of the attribute by restricting the number of characters or figures that can be entered in an input box. The names of the XML elements result from the name of the attribute type, i.e. **<booleanAttribute>**, **<longAttribute>**. Naming conventions that facilitate working with the collected data of the dialog should be used for the IDs of the attributes. That means, when transferring an attribute from a dialog to an attribute of an object, both should be equally named. For example, when creating an object, the attribute of the environment selection in the **client_sign** dialog should have the same name as the corresponding attribute of the object to be created.

EXISTING ATTRIBUTE TYPES

boolean	Attribute for Boolean values (Yes/No).
date	Attribute for date values.
double	Attribute for floating point numbers.
enum	Attribute for enumerations (enumerations_*.xml).
long	Attribute for integers.
selection	Attribute for selecting values from a predefined dynamic list, for example when selecting an environment. PredefinedValueProvider is available to define the possible values of the selection list. The application contains some of these configurable providers (see below). But you can also manually implement some providers.
string	Attribute for simple text input boxes. Content and length of the text can be limited.
text	Attribute for texts consisting of several rows.

There are several **PredefinedValueProviders** for the **selection** attribute. They can be parameterized and provide different possibilities.

EXISTING PREDEFINEDVALUEPROVIDERS

client	<p>Provides a list of all environments that the user has access to. This list can be limited by a maximum of one of the following parameters:</p> <ul style="list-style-type: none"> ▪ componentRight Only outputs environments for which a specific componentRight exists (roles.xml). ▪ objectRight Only outputs environments for which a specific objectRight (roles.xml) exists for the current ObjectType (defined by the objectTypeId parameter). ▪ licensedComponent Only outputs environments that have a specific licensed component. <p>Additional parameters:</p> <ul style="list-style-type: none"> ▪ objectTypeId ID of an object type (objecttypes.xml) that for example, is set by the createObjectDialog command. This parameter is only required in connection with the objectRight parameter.
static	Manages a list of elements defined previously in Java code.
usergroup	<p>Outputs a list with available user groups.</p> <p>Parameter:</p> <ul style="list-style-type: none"> ▪ client_sign Determines which environments the groups should be limited to if clientDependent=true. This parameter can also be defined dynamically by another attribute of the dialog (see the loss_create dialog). ▪ roles A comma-separated list of role names (roles.xml). Only groups with the corresponding roles are considered. ▪ clientDependent Determines whether environment-specific (true) or cross-environment (false) groups are searched for (Default false).
view	Uses a view (views_*.xml) to create a list with elements. The parameters value , id and client require the IDs of the corresponding columns in the view as values.

In the following example a new dialog is created, which is displayed when creating a new control. Some attributes of the control are queried and then set in the newly created control.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Define a new <dialog> element with a unique ID. 2. Add an appropriate subordinate attribute element to each attribute in the dialog. See the text above for possible attribute types. 3. Assign an ID to each attribute type. If the input values should be transferred directly to the attribute of an object, they should both have the same ID, for example client_sign (see also objectTypes.xml). Every dialog and each of its attributes result in the corresponding property keys dialog.<dialogID>.DBI , attribute.<dialogID>.<attributeID>.DBI). 4. Create a new <form> element. The ID of the form is inherited from the dialog ID DIALOG_<dialogID>. Currently such forms consist of exactly one <page> element with a single <rowGroup> element. 5. For each attribute, add a <row> element with an <element> element to the rowGroup. The XML attribute id (row) and attribute.idref (element) refer to the corresponding ID of the dialog attribute. 6. Adapt the corresponding commandchain of the object (see previous chapter).
Documents	dialogs.xsd, forms.xsd, dialogs.xml, forms_*.xml
Example	AddNewInputDialog \WEB-INF\config\custom\xml\custom.xml: Add custom dialog

4.3 Adapt the task configuration

In ARIS Risk & Compliance Manager, transactional objects are accompanied by tasks during their workflow. A task contains the information about which users or user groups are responsible for processing the transactional object. After logging into ARIS Risk & Compliance Manager, users can view their task list under **Home > My tasks**.

The users for processing a task can be specified as follows:

- **One or more directly assigned users**
A list attribute is specified for the transactional object, in which users are directly assigned. By default, tasks for issues are configured in this way.
- **A directly assigned user group**
A list attribute is specified for the transactional object, in which a user group is assigned. The members of this user group are responsible for processing the task. By default, this is the most frequently used configuration. It can be used for configuring the tasks for all transactional objects apart from issues.
- **All environment-specific groups**
A role with the role level **Environment-specific** is specified for the transactional object. The members of the user groups assigned to the role are responsible for processing the task. The transactional object and the user groups must be assigned to the same environment.
If no user group is assigned to the role or no users are assigned to the user group, or if the transactional object is not assigned to any environment, no user is responsible for the task. By default, tasks for deficiencies are configured in this way.
- **All cross-environment groups**
A role with the role level **Cross-environment** is specified for the transactional object. The members of the user groups assigned to the role are responsible for processing the task. If no user group is assigned to the role or no users are assigned to the user group, no user is responsible for the task. By default, this configuration is not used.

Tasks are linked to specific workflow statuses, which determine the subsequent procedure. This set represents the scope of the task and is divided into three statuses per task:

- **Task status Open**
If a transactional object reaches one of the referenced workflow statuses, a new task is started.
- **Task status Completed**
If a transactional object reaches one of the referenced workflow statuses, all tasks in the configuration of that transactional object are classed as completed.
- **Task status Not completed**
If a transactional object reaches one of the referenced workflow statuses, all tasks in the configuration of that transactional object are classed as not completed.

The scope of different task configurations can overlap without causing problems. Thus, for a transactional object with a particular workflow status, there can be several tasks with various responsibilities.

A monitor strategy can be specified for each task. If it is not specified, the task has no due date for the user and retains the status **Open** in the system until, as a result of interaction by the user, the associated transactional object reaches the workflow status referenced by the task status **Completed** or **Not completed**. For example, in the default configuration this is the case for many reviewer tasks.

A monitor strategy specifies the processing and control period for the task. If these periods are not specified, the processing and control period for the transactional object are used. In the default configuration, this is almost always the case if a monitor strategy is specified. One exception are policies, in which the processing period for the task is specified by the **plannedstartdate** and **plannedenddate** attributes, and the control period by the **controlstartdate** and **controlenddate** date attributes.

In the default configuration, this is almost always the case if a monitor strategy is specified. One exception is policies, where the processing period for the task is specified by the **publishingstartdate** and **publishingenddate** date attributes.

All tasks with a monitor strategy are monitored by the monitoring job. The strategy contains monitor levels, which specify what actions the monitoring job should trigger in terms of the task. A monitor level also contains a specified time that determines when the level is classed as reached. This relates to the processing period of the task. The following settings are possible:

- **Percentage**

The monitor level is reached if, at the execution time of the monitoring job, the elapsed processing period has reached or exceeded the specified percentage. The value must be a number between 0 and 100.

Example: Monitor level with Percentage value **50** and processing period 11/1 to 11/30. Monitor level is reached on 11/16.

- **RemainingTime**

The monitor level is reached if, at the execution time of the monitoring job, the remaining time until the end of the processing period is less than or equal to the specified value. The value must be specified using a positive number followed by a letter to represent the time unit. The possible units are **d** (days) or **h** (hours).

Example: Monitor level with RemainingTime value **3d** and processing period 11/1 to 11/30. Monitor level is reached on 11/28.

When a task reaches a monitor level, the monitoring job sends the corresponding message. If no monitor messages are configured, the standard template **monitorjob** is used to send a message to the user responsible for the task. In the default configuration, this is the case for almost all monitor levels. Alternatively, custom messages can be defined for monitor levels. Recipients, CC recipients, the message template and the link to a list in ARIS Risk & Compliance Manager can be specified.

For a monitor level of **Percentage** type with the value **100**, monitor changes can be defined additionally to messages, which use the monitoring job to change values of the transactional object for the task. This can change the object to another workflow status and cause tasks to be created or closed. An example of this in the default configuration is the task configuration for

tester. The task configuration for **issue_owner** is a further example in which an attribute value is changed but this does not have an effect on existing tasks.

If tasks are enhanced by custom attributes, the sources of the inserted values are specified via the XML element `<copyValue>`. The **fromAttribute.idref** attribute and the **toAttribute.idref** attribute of this XML element connect an attribute of the transactional object to an attribute of the generated task. Once the task is generated, the value from **fromAttribute.idref** attribute is copied to **toAttribute.idref** attribute. Both references must link to attributes of the same type. Exceptions: List attributes must link to text attributes. Enumeration attributes are not supported.

Location	XML file in the xml folder
Procedure	Add a new <code><task></code> element at the relevant position. The combination of the id and objectType.idref properties must be unique in the entire configuration. The role.idref property is optional and is only used to display items in the task list. If specified, its value must correspond to the ID of a defined role.
Documents	taskconfiguration_auditmanagement.xml taskconfiguration_changemanagement.xml taskconfiguration_deficiencymanagement.xml taskconfiguration_issuemanagement.xml taskconfiguration_lossmanagement.xml taskconfiguration_policymanagement.xml taskconfiguration_riskmanagement.xml taskconfiguration_signoffmanagement.xml taskconfiguration_surveymanagement.xml taskconfiguration_testmanagement.xml
Example	TaskConfiguration_ModifyLevels\WEB-INF\config\custom\xml\custom.xml: Task with ID tester TaskConfiguration_AddNewColumns\WEB-INF\config\custom\xml\custom.xml : ... All declared tasks

4.4 Adapt a master data import

To assign an attribute from ARIS Architect to an attribute in ARIS Risk & Compliance Manager, you must adapt the mapping file for the ARIS Architect export report accordingly. An attribute mapping always consists of the name in ARIS Risk & Compliance Manager, the attribute type from ARIS Risk & Compliance Manager (see `objectTypes.xml`), and the attribute type from ARIS Architect. This attribute type can name a direct attribute type from ARIS Architect or be a special key word. Customizing offers the following options:

- **<ABA constant name>**: Constant of a default attribute from the ARIS Architect method, e.g., `AT_NAME`.
- **<ABA attribute GUID>**: If the attribute is not a default attribute from ARIS Architect, you can use this GUID.
- **FALSE**: Equivalent to `CONSTANT#false`
- **TRUE**: Equivalent to `CONSTANT#true`
- **DATE_NOW**: Returns the execution time of the report as a value.
- **ISMULTIPLE**: Used for exporting multiEnum attributes. All attributes listed in the respective Enum mapping are searched. The values are read and returned as comma-separated values.
- **CONSTANT#<Constant value>**: Returns the specified constant value.

Simple adaptation of the mapping file is only sufficient for ARIS Risk & Compliance Manager attributes that are not list attributes, because they contain attribute values from ARIS Architect. Apart from adapting the mapping file, a mapping of list attributes also requires rewriting the reports. For a later execution of the export report, you must ensure that the added ARIS Architect attributes are included in the current ARIS Architect filter, otherwise their values cannot be extracted.

Location	XML file <code>aris2arcm_mapping_RBA.xml</code> or <code>aris2arcm_mapping_CBA.xml</code> in ARIS Architect
Documents	<code>aris2arcm_mapping_RBA.xml</code> , <code>aris2arcm_mapping_CBA.xml</code>
Example	<ul style="list-style-type: none"> ▪ <code>ModifyMapping_ABA\WEB-INF\config\custom\xml\custom_mapping_string.xml</code>: Add string attribute ▪ <code>ModifyMapping_ABA\WEB-INF\config\custom\xml\custom_mapping_constant.xml</code>: Add constant number attribute ▪ <code>ModifyMapping_ABA\WEB-INF\config\custom\xml\custom_mapping_single_enum.xml</code>: Add single enum attribute ▪ <code>ModifyMapping_ABA\WEB-INF\config\custom\xml\custom_mapping_multi_enum.xml</code>: Add multi enum attribute

4.5 Add/adapt hierarchies

The following section describes how you add a new hierarchy type and add a simple evaluation using the new hierarchy.

4.5.1 Add an enumeration item

A new hierarchy is added in ARIS Risk & Compliance Manager. A new name is assigned to the hierarchy in the properties. Since the initial root element of the hierarchy is generated in the database only when an environment is generated, you must create a new environment in order to view the change.

Location	XML file in the xml folder
Procedure	Add a new <enumitem> element at the relevant position. The id property contains a unique name for the hierarchy, while the value property contains a unique numerical identifier that is transferred to the database.
Documents	enumerations.xml, enumerations.xsd
Example	AddHierarchy \WEB-INF\config\custom\xml\custom.xml: Step 1

Location	Property file in the properties folder
Procedure	Add a new property at the relevant position and assign a hierarchy name to it.
Example	AddHierarchy \WEB-INF\config\custom\properties\application\custom.properties

4.5.2 Add a new list element to a master data object

To enable evaluation of the hierarchy in the application, the hierarchy must be assigned to an ARIS Risk & Compliance Manager element that has a logical relationship to this new hierarchy. Chapter **Add/adapt a simple attribute** (Page 19) describes how to add new elements.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Add a new <listAttrType> element at the relevant position. 2. In the new listAttrType, insert a <listRestriction> element containing an internal <attributeRestriction> element. 3. You must restrict the hierarchy type to be added to the new attribute.
Documents	objectTypes.xml, objectTypes.xsd
Example	AddHierarchy \WEB-INF\config\custom\xml\custom.xml: Step 2

Location	Property file in the properties folder
Procedure	Add a new property at the relevant position and assign the name of the new list attribute to it.
Example	AddHierarchy \WEB-INF\config\custom\properties\application\custom.properties

4.5.3 Add a new list element to a transactional object

The relationship to the new hierarchy is now to be transferred to the transactional object that object owners are using and whose status can be evaluated using the hierarchy. Chapter **Add/adapt a simple attribute** (Page 19) describes how to add new list elements.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Add a new <listAttrType> element at the relevant position. 2. In the new listAttrType, insert a <listRestriction> element containing an internal <attributeRestriction> element. 3. Restrict the hierarchy type to be added to the new attribute.
Documents	objectTypes.xml, objectTypes.xsd
Example	AddHierarchy \\WEB-INF\\config\\custom\\xml\\custom.xml: Step 3

Location	Property file in the properties folder
Procedure	Add a new property at the relevant position and assign the name of the new list attribute to it.
Example	AddHierarchy \\WEB-INF\\config\\custom\\properties\\application\\custom.properties

4.5.4 Display and input options for forms

You can import the data of the new hierarchy from ARIS Architect because the structure is usually specified there. However, the example shows how to manually create an input option in the form of the master data object. Chapter **Add an attribute to a form** (Page 21) describes how to add new list elements to forms.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Add a new <row> element at the relevant position. 2. In the new row, insert an <element> element containing the internal <parameter> element. 3. Set the selection list to AUTO. This enables you to select a new hierarchy element without having to create an additional selection list.
Documents	forms_testmanagement.xml, forms.xsd
Example	AddHierarchy \\WEB-INF\\config\\custom\\xml\\custom.xml: Step 4 and Step 5

You can also display this new attribute in lists. See **Add an attribute to a list** (Page 23).

4.5.5 Automatic transfer of hierarchy objects

If the attribute names comply with the conventions, meaning they are identical for master data object and transactional object, the generator automatically transfers the hierarchy object (in the example: during test case generation). The conventions are described in the section **Conventions for object generation** (Page 11).

4.5.6 Make a hierarchy attribute editable.

Section **Add/adapt rule** (Page 71) describes how to edit rules.

Location	DRL file in the rules folder.
Procedure	Enable editing the attribute in the master data object in the rule Define all standard attributes as editable .
Documents	risk.drl
Example	AddHierarchy \WEB-INF\config\custom\rules\risk.drl: Enable the editing of a hierarchy attribute in rules

4.5.7 Assign roles to a hierarchy attribute

You can add a hierarchy element to a role or remove it. Section **Adapt roles (assign/remove)** (Page 30) describes how to edit roles.

Location	XML file in the xml folder
Procedure	In the <relation> element, connect the list attribute to an Assign and/or Remove privilege.
Documents	roles.xsd
Example	AddHierarchy \WEB-INF\config\custom\xml\custom.xml: Step 6

4.5.8 Add a hierarchy evaluation

Location	XML file in the xml folder
Procedure	Add a new <nav.evaluation> element and connect it to the existing statistics definition.
Documents	navigation_evaluation.xml, navigations.xsd, evaluations.xml, evaluations.xsd
Example	AddHierarchy \WEB-INF\config\custom\xml\custom.xml: Step 9

4.5.9 Create a new data view for hierarchy statistics

Assign a new data view to the statistics. Section **Adapt statistics** (Page 57) describes how to edit statistics views.

Location	XML file in the xml folder
Procedure	In the <view> element, create a new data view that connects the new hierarchy to the transactional object to be evaluated and to the user group.
Documents	views.xsd
Example	AddHierarchy \WEB-INF\config\custom\xml\custom.xml: Step 7

You must also create a new view that generates the linked lists of the connected transactional objects from the statistics.

Location	XML file in the xml folder
Procedure	In the <view> element, create a new data view that generates the linked lists of the connected transactional objects from the statistics.
Documents	views.xsd
Example	AddHierarchy \WEB-INF\config\custom\xml\custom.xml: Step 8

4.6 Add/adapt statistics

Statistics are evaluations that show the distribution of data across a structure. For example, the test case statistics shows the distribution of test results across a hierarchy.

4.6.1 Adapt statistics

Use the XML configuration to perform the following actions (sorted by complexity) for statistics:

- Adapt the column width
- Link structural elements to associated forms
- Add/adapt column(s)
 - `statistic.columnGroup.enum-based` header
 - `statistic.columnGroup.perCent-based` header
 - `statistic.column.value-based` header
- Adapt links
- Incorporate new hierarchy

4.6.1.1 Adapt column widths

The width attribute is optional for all columns. If the column width is not configured, the width of the individual columns is calculated automatically. By default, a width of 20% is assigned to the structure, and a width of 1% is assigned to each chart column. The remaining percentage is equally distributed to the other columns.

Since Excel and PDF reports do not contain chart columns, the layout is recalculated for a report, i.e., the width of the chart columns is distributed to the columns visible in the report.

4.6.1.2 Link structural elements

The tree view of the statistics usually contains nodes that represent objects, such as hierarchies or user groups. You can link these objects to the nodes of the tree view. Linked objects are displayed as a pop-up window.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none">1. Copy the <evaluation> element from the default configuration to the customizing file.2. Add a linkedNodeTypes attribute to the <statistic.column.tree> element. A comma-separated list of the object type IDs must exist as an attribute value.
Remark	A structural element representing an object type of the comma-separated list is displayed as a link. The link opens the associated object in a pop-up window.
Documents	evaluations.xsd, evaluations_*.xml
Example	ModifyStatistic \WEB-INF\config\custom\xml\evaluation_custom.xml: Add custom linked nodes

4.6.1.3 Add/adapt columns

4.6.1.3.1 `statistic.columnGroup.enum`-based statistics

Evaluations showing the percentage distribution of enumeration attribute values of specific objects are automatically extended by a column if the corresponding enumeration is extended.

Location	XML file in the <code>xml</code> folder
Procedure	<ol style="list-style-type: none"> 1. Copy the <code><enum></code> element from the default configuration to the customizing file. 2. Create one or more new <code><enumitem></code> elements within the <code><enumitem></code> element. Within the <code><enumitem></code> element, you can define the color of the column header using the <code><parameter></code> element. The same color is used for the display in pie charts. <p>Example: <code><parameter name="Background" value="EBB585"/></code></p>
Remark	<p>This only applies to statistics whose columns are configured using the <code><statistic.columnGroup.enum></code> XML element. Virtual enumeration items are used for structuring the column header. This means that real items are subordinate to virtual items in the header.</p> <p>You can prevent columns representing enumeration attribute values (real or virtual) from being displayed by inserting the <code>evaluationRelevant</code> attribute with the value <code>false</code>.</p>
Documents	<code>evaluations.xsd</code> , <code>evaluations_*.xml</code>
Example	<p>ModifyObjectLifecycle</p> <p><code>\WEB-INF\config\custom\xml\testcase_enum_custom.xml</code>: New item as a configuration example</p>

4.6.1.3.2 `statistic.columnGroup.perCent`-based statistics

This element is used for showing absolute and percentage values for non-enumeration values.

Location	XML file in the <code>xml</code> folder
Procedure	<ol style="list-style-type: none"> 1. Copy the <code><evaluation></code> element from the default configuration to the customizing file. 2. Adapt a <code><statistic.columnGroup.perCent></code> element. You can add the individual columns of the group using <code><statistic.column.value></code> elements. If your calculation is based on <code><statistic.calculator id="value"/></code>, you can use the value of the <code>calculation</code> attribute to control the calculation of the column values.
Remark	<p>The <code>calculation</code> attribute can have the following values:</p> <p>count: The value of the column is increased by 1 for each value found in the data source.</p> <p>sum: All values of the data source are added up. The values of the data source must be of the Number type.</p> <p>By default, each column group contains a balance column and a pie chart. The balance column contains the sum of values of all columns that are configured using the <code><statistic.column.value></code> element. Each sector of a pie chart represents a configured column.</p>
Documents	<code>evaluations.xsd</code> , <code>evaluations_*.xml</code>
Example	<p>ModifyStatistic</p> <p><code>\WEB-INF\config\custom\xml\evaluation_custom.xml</code>: Add custom value column <code>perCent</code></p>

4.6.1.3.3 `statistic.column.value`-based statistics

Individual values that are not part of a group of percentage values can be added or adapted using the `<statistic.column.value>` element. These values are then displayed as absolute values.

Location	XML file in the <code>xml</code> folder
Procedure	<ol style="list-style-type: none"> 1. Copy the <code><evaluation></code> element from the default configuration to the customizing file. 2. Adapt a <code><statistic.column.value></code> element. If your calculation is based on <code><statistic.calculator id="value"/></code>, you can use the value of the <code>calculation</code> attribute to control the calculation of the column values.
Remark	<p>The <code>calculation</code> attribute can have the following values:</p> <p>count: The value of the column is increased by 1 for each value found in the data source.</p> <p>sum: All values of the data source are added up. The values of the data source must be of the Number type.</p>
Documents	<code>evaluations.xsd</code> , <code>evaluations_*.xml</code>
Example	<p>ModifyStatistic</p> <p><code>\WEB-INF\config\custom\xml\evaluation_custom.xml</code>: Add custom value column</p>

4.6.1.3.4 Adapt links

Each cell, or the values displayed therein can be linked. Usually, the link represents the value of the cell. Example: If the value in a cell of a column that shows the number of open test cases is 10, the link will open a list with 10 open test cases.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy the <evaluation> element from the default configuration to the customizing file. 2. Adapt or add the link.typ and link.idref attributes of the <statistic.column.value>, <statistic.columnGroup.perCent>, and <statistic.columnGroup.enum> elements. list and evaluation are permitted as a link.typ attribute value. An ID from the file lists_*.xml or evaluations_*.xml is expected as a link.idref attribute value.
Remark	The linked list or evaluation is filtered using the values that were used for filtering the data source of the statistics. This means that the view of linked list or evaluation is subject to the same conditions.
Documents	evaluations.xsd, evaluations_*.xml, lists_*.xml
Example	ModifyStatistic \WEB-INF\config\custom\xml\evaluation_custom.xml: Add custom linked column

4.6.1.3.5 Use a new hierarchy

For details on using new hierarchies, see the chapter **Add/adapt hierarchies** (Page 52).

4.7 Add/adapt reports

Report definitions can be customized in various ways. Adaptations that were realized only through changes to the XML configuration are only possible for form and list reports. They are described in the two following chapters.

4.7.1 Add/adapt reports for forms

Specialized report definitions already exist for some forms, such as the questionnaire. They are missing, however, for other objects, such as risk and control. In this case, a report definition is automatically generated from the form and immediately applied before report execution starts. For forms and lists, these report definitions are written to the file **reports_dynamicreports.xml** so that they can be used as customizing examples. The following examples explain how existing report definitions for forms can be replaced, and how new report definitions can be defined for forms that have no specialized report definition yet.

4.7.1.1 Replace an existing form report definition

To replace an existing form report, simply specify a report definition with the same ID and format in any customizing XML file. This report definition is used instead of the default definition. In the example below, the default form report of the questionnaire is replaced by a shorter version, from which the output of questions is excluded.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy the report definition of the form. 2. Change it according to your requirements.
Documents	Report XML files in the default configuration serving as templates
Example	FormReports_ReplaceExisting \WEB-INF\config\custom\xml\custom.xml: Replace standard questionnaire form report with a simplified version

4.7.1.2 Add a new form report definition

The procedure of adding a form report to forms without a specialized report definition is the same as the procedure described in the above example. Ensure that the report ID corresponds to the name of the form object type and is written in upper-case letters. In the following example, a PDF form report for risks is defined. Therefore, the report ID is **RISK**. It references the subordinate form report **CONTROL**. As this report is not defined explicitly, ARIS Risk & Compliance Manager continues to use a definition directly generated from the form.

Location	XML file in the xml folder
Procedure	Create a new report definition.
Documents	Report XML files in the default configuration serving as templates
Example	FormReports_AddNew \WEB-INF\config\custom\xml\custom.xml: Add a new risk form report

4.7.1.3 Incorporate a new form report selection

Instead of creating a form report, you can configure a selection dialog that references various other reports. The ID of such a selection dialog must either correspond exactly to the form object type in capital letters or have the suffix **_SELECT**. The following example defines a report selection for an EXCEL form report of a risk assessment allowing you to additionally select the default form report, a special user-defined report, or a combination of both. For the Excel format, the default form report is created automatically.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Create a new report definition. 2. Enable Excel report in the form.
Documents	Report XML files in the default configuration serving as templates
Example	<ul style="list-style-type: none"> ▪ FormReports_AddSelection\WEB-INF\config\custom\xml\custom_report.xml : Add a new selection ▪ FormReports_AddSelection\WEB-INF\config\custom\xml\custom_report.xml : Add a new form report ▪ FormReports_AddSelection\WEB-INF\config\custom\xml\custom_forms.xml: Activate excel report button

4.7.2 Add/adapt reports for lists

Specialized report definitions already exist for some lists, such as the user list or test case list. They are missing, however, for other lists, such as risk and control lists. In this case, a report definition is automatically generated from the relevant list and immediately applied before report execution starts. For forms and lists, these report definitions are written to the file **reports_dynamicreports.xml** so that they can be used as customizing examples. The following examples explain how existing report definitions for lists can be replaced, and how new report definitions can be defined for lists that have no specialized report definitions yet.

4.7.2.1 Replace an existing list report definition

To replace an existing list report, simply specify a report definition with the same ID and format in any customizing XML file. This report definition will be used instead of the default definition. In the example below, the default list report for users is replaced by a shorter version that reduces the number of columns.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy the report definition of the list. 2. Change it according to your requirements.
Documents	Report XML files in the default configuration serving as templates
Example	ListReports_ReplaceExisting \WEB-INF\config\custom\xml\custom.xml: Replace standard user list report with a simplified version

4.7.2.2 Add a new list report definition

The procedure of adding a list report to lists without a specialized report definition is the same as the procedure described in the above example. The report ID must be identical to the list ID. In the following example, a PDF list report is defined for the list of risks in Explorer, which is displayed for the test manager. Therefore, the report ID is **RISK**.

Location	XML file in the xml folder
Procedure	Create a new report definition.
Documents	Report XML files in the default configuration serving as templates
Example	ListReports_AddNew \WEB-INF\config\custom\xml\custom.xml: Add a new risk list report

4.7.2.3 Incorporate a new report selection

Instead of creating a list report, you can configure a selection dialog that references various other reports. The ID of such a selection dialog must either correspond exactly to the list ID or must have the suffix **_SELECT**. The following example defines a report selection for the Excel list report of the risk assessment list in Explorer allowing you to additionally select the default list report, a special user-defined report, or a combination of both. For the Excel format, the default list report is created automatically.

Location	XML file in the xml folder
Procedure	Create a new report definition.
Documents	Report XML files in the default configuration serving as templates
Example	<ul style="list-style-type: none">▪ ListReports_AddSelection\WEB-INF\config\custom\xml\custom.xml: Add a new selection▪ ListReports_AddSelection\WEB-INF\config\custom\xml\custom.xml: Add a new list report

4.8 Modify message template

4.8.1 Add a new message template

To be able to use a new message template you must add it first.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy the <enum> element with the ID initiators from the default configuration to customizing. 2. Next, use the <enumitem> element to create new message templates within the <enum> element. You must at least specify the attributes id and value. The value must be unique for both attributes within the <enum> element. 3. Create a new property entry for the new message template (see Add/adapt properties of an enumeration (Page 26)). This entry enables the display of the message template in the list of templates.
Documents	enumerations.xml, enumerations.xsd
Example	<ul style="list-style-type: none"> ▪ ModifyMessageTemplate_AddNewMessageTemplate\WEB-INF\config\custom\xml\custom.xml: Add new message template ▪ ModifyMessageTemplate_AddNewMessageTemplate\WEB-INF\config\custom\properties\application\custom.properties: Add new message template to template list


4.8.2 Add a new message template content

After adding a new message template, you must add contents to it.

Procedure	Add two new property entries to the custom.properties file (see Add/adapt properties (Page 19)).
Remark	The names of the new entries must comply with the following conventions: Subject of the message: message.<template_name>.subject.DBI Contents of the message: message.<template_name>.text.DBI
Example	ModifyMessageTemplate_AddNewMessageTemplate \WEB-INF\config\custom\properties\application\custom.properties: Add contents to new message template

4.8.3 Customize the contents of a message template

You can adapt the contents of a message template in ARIS Risk & Compliance Manager.

<p>Procedure</p>	<ol style="list-style-type: none"> 1. Click  Administration. 2. Under System management, click Environments or System depending on which message template you want to edit. 3. Open the message template you want to display or change. The corresponding text is displayed in the Message field. 4. Change the text in the template and subject as required. 5. If you want to send the text from a different language version of ARIS Risk & Compliance Manager, select the relevant language in the Language box and enter the message text in the Template box. 6. Use the Notification type field to specify whether notifications are to be sent by e-mail and/or to the internal ARIS Risk & Compliance Manager mailbox. <p>Once you saved your changes, the contents of the message template is saved to the database. As long as you do not restore the message template to the default via the user interface, the content from the database is used. Another method of modifying a message template is similar to the method of adding a new message template (see Add a new message template (Page 67)). The only difference is that you adapt the property value.</p>
<p>Remark</p>	<p>If a message template requires values that can only be determined dynamically during runtime, they are incorporated in the form of objects and variables. The difference between both options is that an object provides the methods for accessing values, while a variable has a given value.</p> <p>By default, the object responsible for the form flow is always available. You can access all attributes defined for these object types including Inheritance (Page 4) via the corresponding method.</p> <p>The usage of objects in a message template must comply with the following conventions:</p> <ul style="list-style-type: none"> ▪ For an object or a variable to be accessible, the prefix \$ must always be added. ▪ The object name always corresponds to the defined attribute ID of the <objectType> element in lower-case letters. ▪ The method name always consists of the prefix get and the attribute ID of the <attrType> element. Note that get must always be followed by an upper-case letter. Furthermore, underscores in the ID are identified via a filter and replaced with the upper-case letter of the subsequent word.

	<p>Example</p> <p>Object type Testdefinition with defined ID TESTDEFINITION and attribute owner_group</p> <p>Access to this attribute in the message template: \$testdefinition.getOwnerGroup()</p> <p>The following additional objects and variables are available by default:</p> <ul style="list-style-type: none"> ▪ \$user – Object providing recipient information ▪ \$client – Object providing environment information ▪ \$serverConnection – Variable providing a link to the server <p>If more objects or variables are required, they can only be provided by Java code implementation. User-defined implementations are indicated by message templates that already use additional objects and variables.</p>
Documents	objectTypes.xml, objectTypes.xsd
Example	ModifyMessageTemplate_AddNewMessageTemplate \WEB-INF\config\custom\properties\application\custom.properties: Add contents to new message template

4.8.4 Send messages

Once a new message template was created, it can be incorporated into the form flow and sent.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy a <catalog> element from the default configuration to the customizing file. 2. Use the <command> sendMail element to add the new message template to the <commandchain> element. 3. Define the new message template and the message recipients using <parameter> elements. In this context, usage of the parameter cc is optional. The defined id attribute of the associated <listAttrType> element - which additionally contains the objectType.idref attribute with the values USER or USERGROUP - is always a permitted value for the parameters to and cc.
Documents	commandchains_[module].xml, commandchains.xsd, objectTypes.xml, objectTypes.xsd
Example	ModifyMessageTemplate_AddNewMessageTemplate \WEB-INF\config\custom\xml\custom.xml: Add send mail command to send new message

4.9 Add/adapt segregation of duties

Segregation of duties is to ensure that specific objects are subject to dual control. For this, you must configure a pair of roles users must not have simultaneously if they want to edit these objects. For example, a test case must not be processed by a user who has the **Tester** and the **Reviewer** role for this test case.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy the <segregationsOfDuties> element from the default configuration to the customizing file. 2. Create one or more <segregationsOfDuties> elements within the <segregationsOfDuties> element. 3. In the objectType attribute, specify the object type of the object for which a segregation of duties is to be configured. 4. Within the <segregationOfDuties> element, specify the two roles users must not have concurrently if they want to edit an object in two separate <segregationsOfDuties> elements.
Remark	To revoke dual control, remove the <segregationsOfDuties> element.
Documents	segregationsOfDuties.xsd, segregationsOfDuties.xml
Example	<pre> <segregationOfDuties objectType="testcase"> <segregationsOfDuties.role id="tester"/> <segregationsOfDuties.role id="testreviewer"/> </segregationOfDuties> </pre>

4.10 Add/adapt rule

Rules are used to manipulate the behavior of forms. In the default configuration, the rules that are to be applied to a form are stored in individual DRL files named for the object type of the form (for example, risk.drl).

4.10.1 Overwrite an existing rule file

To overwrite an existing rule file, save a file with the same name to the path **WEB-INF/config/custom/rules**. When the server is started, this file is referred to instead of the default file. All conditions and results defined in rules are textual descriptions of the functionality that is to be implemented by the rule. They must be DSL items specified in the DSL file, which in turn is referenced in the DRL file as well as in its defined rule set. The example shows how to add a new attribute to the **USERGROUP** object, how to extend the form accordingly, and how to replace existing form rules with an extended version. It also illustrates the definition of a new attribute and adaptation of the form.

Location	DRL file in the rules folder.
Procedure	Copy the DRL file of the required form to the folder mentioned above. If necessary, change existing rules or add new rules to the file.
Documents	Corresponding DSL file in the default configuration Java doc of the CollectiveHelper class and its derived classes for an overview of possible conditions and consequences.
Example	ModifyRules_ReplaceDRL\WEB-INF\config\custom\rules\usergroup.drl: Add custom rule ModifyRules_ReplaceDRL\WEB-INF\config\custom\xml\custom.xml: Add new usergroup attribute creator_remark ModifyRules_ReplaceDRL\WEB-INF\config\custom\xml\custom.xml: Add text field in usergroup form for new usergroup attribute 'creator_remark'

4.10.2 Incorporate a new rule file

To incorporate a new rule file, you must define a new rule set that must reference the same DSL file that is referenced within the rule file. This rule set must be saved to a custom XML file under **WEB-INF/config/custom/xml**. Furthermore, you must overwrite the existing rule context to which the new rule set is to belong, and also save it to a custom XML file under the path mentioned above. In the following example, the rules of the USERGROUP form are extended by two new rules.

Location	DRL file in the rules folder XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy the DRL file of the required form. 2. Keep the header up to the import statements and delete the rest. 3. Add new rules according to your requirements. 4. Incorporate the new DRL file into a new rule set and save it to a custom.xml file. 5. Overwrite the rule context of the corresponding form and incorporate the new rule set in addition to the default set.
Documents	<p>Corresponding DRL file in the default configuration</p> <p>Corresponding DSL file in the default configuration</p> <p>rulesetReg.xml in the default</p> <p>Java doc of the CollectiveHelper class and its derived classes for an overview of possible conditions and consequences.</p>
Example	<p>ModifyRules_AddDRL\WEB-INF\config\custom\rules\usergroup.drl: Add custom rule set</p> <p>ModifyRules_AddDRL\WEB-INF\config\custom\xml\custom_ruleContext.xml: Overwrite the ruleContext and add the custom ruleSet</p> <p>ModifyRules_AddDRL\WEB-INF\config\custom\xml\custom_ruleContext.xml: Add custom ruleSet</p> <p>ModifyRules_AddDRL\WEB-INF\config\custom\xml\custom_usergroupform: Enable rule execution on change of form element 'name'</p>

4.10.3 Reuse existing rules for new attributes

To add a new attribute with identical behavior in terms of visibility and mandatory field conditions to an existing group of attributes, you can assign the **behavesLike** characteristic to the object type. The following example shows this mechanism with the risk owner being provided with a new mandatory attribute for the case that the risk assessment is to be assessed with the **Qualitative** type.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy the definition of the relevant object type and add the new attribute. 2. Under behavesLike at this attribute, enter the name of another defined attribute the behavior of which the new attribute is to adopt. 3. Add the attribute to the form.
Documents	objectTypes.xml in the default
Example	ModifyRules_UseFreeriders \WEB-INF\config\custom\xml\custom.xml: Add custom freerider attribute ModifyRules_UseFreeriders \WEB-INF\config\custom\xml\custom.xml: Add new attribute into risk assessment form

4.11 Add/adapt a scheduled task

4.11.1 Adapt the schedule

Generally, ARIS Risk & Compliance Manager provides two options for starting automated jobs in order to generate or change objects. You can trigger a job manually via the ARIS Risk & Compliance Manager interface if you have the corresponding manager role. Or the job is executed according to schedule by ARIS Risk & Compliance Manager. For example, in the default configuration this applies to generator and monitoring jobs. Scheduled jobs are configured by ACC commands.

While most of the configuration parameters are simple key value pairs, the configuration of job scheduling is more complex. Job configuration parameters always start with **arcm.config.schedule.job**, followed by the job type (generator, monitor, updater), and then by the object type for which the job is responsible. The cleaning job does not have this object type appendix. The value of such a job parameter is built up as a sequence of key value pairs, grouped by enclosing braces `[]`. Key and value are separated by the pipe character (`|`).

Example:

```
reconfigure arcm_m arcm.config.schedule.job.generator.testcase=[ jobitem |
generatorJob ] [ startScheduler | true ] [ executionTime | 0 52 00 ? * SUN-SAT ] [
clientexcludinglist | ] [ clientincludinglist | ] [ objecttypes | TESTCASE ]
```

The individual parameters have the following meaning:

- **Jobitem**
The job to be executed. The parameter value must correspond to an EnumItem ID from the **jobs** enumeration in the **enumerations.xml** file.
- **startScheduler**
Must be **true** so that the time control for this job is active.
- **executionTime**
This expression states at which point in time the job should be started. It has the format **CronTrigger** that allows the specification of time intervals.

The individual values mean the following from left to right:

- **Seconds** (0-59)
- **Minutes** (0-59)
- **Hours** (0-23)
- **Day of month** (1-31)
- **Month** (1-12 or JAN-DEC)
- **Day of week** (1-7 or SUN-SAT)
- **Year** (can be empty, 1984, 1970-2099, ...)

In the example above, the monitor job for checking test cases on each day of each month starts at 01:52. For further information, see the CronTrigger documentation on the Quartz home page (<http://www.quartz-scheduler.org> (<http://www.quartz-scheduler.org>))

- **excludedEnvironments**

The environments in the ARIS Risk & Compliance Manager database for which the job should not be executed are listed here. The values can be specified separated by commas. This applies to environment specific jobs only.

- **includedEnvironments**

The environments in the ARIS Risk & Compliance Manager database for which the job should be executed are listed here. If no value is specified, a separate job is started for each individual environment. The values can be specified separated by commas. This applies to environment specific jobs only.

- **Objecttypes**

The object types for which the job should be executed. In the example above, this instance of the monitoring job should only check the test cases. The values can be specified separated by commas.

4.11.2 Generator

4.11.2.1 Adapt the object search

The search for initial recurring objects, such as a test definition for test cases or a risk for risk assessment is controlled in the default configuration via the file

commandchains_generator.xml. It includes a **generator_[target type]** chain, which is linked to either the **RecurringObjectSearchCommand** command or a derivation of it.

This command is responsible for finding all recurring objects that come into consideration for the generation of transactional objects. These objects or rather their OVIDs are written by Command in the CommandChainContext and from there extracted and reused by the generator.

Location	XML file in the xml folder
Procedure	Copy the appropriate <catalog> from the default configuration to customizing. Then, adapt the command chain.
Documents	commandClassMapping.xml, blClassMapping.xsd, commandchains_generator.xml, commandchains.xsd

4.11.2.2 Generate objects

The generator creates an empty transactional object for each OVID of a recurring object that the generator receives by performing the object search. Whether this object can be filled and saved correctly is determined by the workflow configuration of the transactional object.

Each transactional object type in the workflow has a connection in **<state.initial>** for which only the generator jobs are authorized (**permission**). When the object is created, the generator follows this workflow connection.

Example for the TESTCASE object type

Command chain **prepareJob** in the **testcase** catalog.

The chain with which the transactional object is checked and filled is defined on this workflow connection. The first part of the chain consists of derivations of the **GeneratorConditionCheckCommand** class, each of which checks for the relevant recurring object (e.g., test definition) whether all conditions for the generation are met. The second part consists of a derivation of **GenerateCommand**. Here attribute values are assigned to the new transactional object, e.g. test case. Since these are commands, the behavior can be adapted by omitting, replacing, and supplementing.

After the chain described above was successfully completed, the transactional object is in the **<state.prepared>** status. This means that the generator creates the object, but the object is not yet persistent. In the second step, the generator follows the connection to which it is authorized (permission) and which leads to the first workflow status that allows manual processing by the user. In the default configuration, these are the states in which the tasks for the owner or the creator are defined.

Example for the TESTCASE object type

Command chain **insertJob** in the **testcase** catalog.

The command **prepareJobMessageCommand** within this second connection defines which notifications should be sent to the owner. During its cycle, the generator collects the messages from all of the created objects, combines them as far as possible and sends them to the recipients defined above.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy the appropriate <catalog> from the default configuration to customizing. 2. Adapt the command chain.
Documents	<ul style="list-style-type: none"> ▪ commandClassMapping.xml, blClassMapping.xsd, ▪ commandchains_[module].xml, ▪ commandchains.xsd, workflow_[module].xml

4.11.3 Adapt the object search

The monitoring job can be adapted so that for transactional objects, e.g., test cases, you can specify in their workflow whether they are to be checked by the monitoring job or not. If an object with a specific workflow state is to be checked by the monitoring job, a task item must be entered for the state. In the default configuration, these are the states that represent the processing carried out by the owner group.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy the appropriate workflow to customizing. 2. Modify the <task.item> elements at the individual states.
Remark	Make sure that the time.limitation property in the task definition is not set to false , otherwise, the monitoring job ignores these tasks.
Documents	workflow.xsd, workflow_[module].xml

4.11.4 Updater

Location	XML file in the xml folder
Procedure	See Generator (Page 76)
Documents	Commandchain_update.xml

4.12 Adapt offline processing

Offline processing cannot be customized in general. This applies to downloading in forms and lists, assignment of offline processors, generation of one or more offline documents, uploading with manual or automatic confirmation, etc. However, up to a certain point, the predefined steps can be modified in terms of content. The following chapters describe the various options as XML configuration or program adaptations.

4.12.1 Modify offline documents

The default implementation of offline processing uses the report engine to generate special Excel documents during the download or to import them during the upload. This procedure is based on specific report definitions stored in default report XML files called **reports_offlineprocessing_<component name>.xml**. These report definitions must adhere to additional conventions:

- Their ID must always be specified in upper-case letters. Example: **<ID of object type>_<ID of editor role>**.
- The default implementation uses Excel reports only, which means that the required format must be Excel.
- Excel cells that offline editors can to edit must be marked with the renderer **offlineProcessingInputReferenceRenderer** or, optionally, with the style **offlineinputcell**.
- The auto component **offlineinfo** must be part of the report definition.

When adapting editable cells in the report definition, you must ensure that these cells are marked as editable by the rule engine for the selected editor role. Otherwise, all attempts to upload documents are canceled. The following example shows how to incorporate the test case attributes **Walkthrough name** and **Walkthrough counter** into offline processing for the tester.

Location	XML file in the xml folder DRL file in the rules folder.
Procedure	<ol style="list-style-type: none"> 1. Copy the TESTCASE_TESTER report definition of the form to a separate custom.xml file and use the offlineProcessingInputReferenceRenderer report renderer for the cells of the Walkthrough name and Walkthrough counter attributes. 2. Copy the rules of the test case form to a separate testcase.drl file. 3. Adapt the rules so that the tester is allowed to edit the two attributes in the form.
Documents	reports_offlineprocessing_testmanagement.xml as a template testcase.drl as a template
Example	<ul style="list-style-type: none"> ▪ OfflineProcessing_CustomizeDocument\WEB-INF\config\custom\xml\custom.xml: Enable offline processing for the two attributes in the offline document ▪ OfflineProcessing_CustomizeDocument\WEB-INF\config\custom\rules\testcase.drl: Mark the two attributes as editable for testers

4.12.2 Change the offline operator roles definition

The **offlineProcessing.xml** configuration file specifies the offline editor roles that can occur as operators of other roles. In the default configuration, these operators are the manager groups for **Test Management**, **Risk Management**, and **Survey Management**, which can start a check-out for their owner and reviewer groups. Adding a new role, whether intended for offline processing or not, always requires comprehensive customizing of ARIS Risk & Compliance Manager and the workflows. Therefore, the example below only shows how to restrict the Operator role of the risk manager because this does not require the creation of a new role. In this example, the risk manager functions as an operator only for the risk owner, and no longer for the risk reviewer.

Location	XML file in the xml folder
Procedure	Copy the <offline-operators> XML element from the file offlineProcessing.xml . Keep only the <operator-role> entry and change it.
Documents	offlineProcessing.xml as a template
Example	OfflineProcessing_ChangeOperator\WEB-INF\config\custom\xml\custom.xml: Restrict operator role of risk manager

4.12.3 Add a new Offline editor role

The **offlineProcessing.xml** configuration file specifies the roles existing in ARIS Risk & Compliance Manager which can act as Offline editor roles for specific objects. In the default configuration, these roles for **Test Management**, **Risk Management**, and **Survey Management** can be assumed by the relevant manager, owner, and reviewer groups. Adding a new role always requires comprehensive customizing of ARIS Risk & Compliance Manager and the workflows. Therefore, the XML fragment below (extract from a custom.xml) only shows how to add a new **testvalidator** role without any further integration into the existing workflows.

```
<offline-editable>
<object-type name="testcase">
<object-type-role id="testmanager"/>
<object-type-role id="tester"/>
<object-type-role id="testreviewer"/>
<!-- enable new role for offline processing -->
<object-type-role id="testvalidator"/>
</object-type>
...
</offline-editable>
```

4.12.4 Adapt offline processors

The example **Modify offline documents** (Page 79) shows how to adapt Excel documents from offline processing, which were created by the report engine. It is also possible to generate Excel files in a completely different way or use other document formats by replacing the default offline processors by a custom implementation that generates such documents, extracts changes made by the offline editor, and re-imports the changes into ARIS Risk & Compliance Manager. These classes must implement the **ICheckOutProcessor** or **ICheckInProcessor** interface and can then be incorporated into a **custom.xml** via the XML fragment below:

```
<processors>
  <checkOut>
    <checkOutProcessor
format="PDF"
clsName="com.idsscheer.webapps.arcm.bl.offlineprocessing.processors.MyCustomPDFC
heckOutProcessor"/>
  <checkIn>
    <checkInProcessor format=" PDF "
clsName="com.idsscheer.webapps.arcm.bl.offlineprocessing.processors.
MyCustomPDFCheckInProcessor "/>
  </checkIn>
</processors>
```

4.12.5 Adapt offline behavior for each object type

For each object type, the implementation of the **IOfflineProcessingBehaviour** interface assigned to it determines the circumstances under which a specific object is classified as offline editable depending on its state and current user. In the default configuration, a special feature of the questionnaire is, for example, that all subordinate sections, questions, etc., are locked for editing once offline processing starts for the questionnaire. You can customize this behavior by incorporating custom implementations of the above-mentioned interfaces in a **custom.xml** file using the following XML fragment:

```
<processingBehaviour>
  <controller objectType="testcase"
clsName="com.idsscheer.webapps.arcm.bl.offlineprocessing.behaviour.custom.MyTCOf
flineProcessingBehaviour"/>
</processingBehaviour>
```

4.13 Add/adapt dashboard link

4.13.1 Adapt Dashboard link

If you adapt an object type for customizing, e.g., if you add an attribute, you are recommended to adapt the existing MashZone list accordingly. For this, copy the relevant MashZone data query to the customizing area and modify it as required.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none">1. Insert a copy of the <view> element from the default configuration below the <custom> element.2. Modify as required, e.g. insert additional attributes or remove attributes.
Documents	mashzone_views.xml, views.xsd
Example	<ul style="list-style-type: none">▪ AddModifyMashzoneURL\WEB-INF\config\custom\xml\custom.xml: Modify data list▪ AddModifyMashzoneURL\WEB-INF\config\custom\properties\application\custom.properties: Modify data list▪ AddModifyMashzoneURL\WEB-INF\config\custom\rules\issue.drl: Modify data list

4.13.2 Add dashboard link

4.13.2.1 Add a MashZone list for object data

To add a new list to ARIS Risk & Compliance Manager in **Administration > Integration > Generate dashboard link**, you must create a new MashZone-relevant data query (**view**) in the configuration. See the associated XML schema **views.xsd** for details. The list and filter view is generated automatically in the link generator if the **relevantForMashzoneIntegration** attribute of the relevant **<view>** element is set to **true**. In this case, you can also run a CSV query via the MashZone interface.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Create a new <view> element below the <custom> element. 2. MashZone data queries are subject to the same rules as the other data queries. However, they must be marked with the flag relevantForMashzoneIntegration="true". <p>For example, this enables you to quickly convert existing data queries into MashZone data queries by copying them and setting the true flag for them. All columns (<viewsColumn>) that are not explicitly excluded by setting the meshzoneRelevant="false" flag are available in the URL generator later.</p>
Documents	mashzone_views.xml, views.xsd
Example	<ul style="list-style-type: none"> ▪ AddModifyMashzoneURL \WEB-INF\config\custom\xml\custom.xml: Add new data list ▪ AddModifyMashzoneURL \WEB-INF\config\custom\properties\application\custom.properties: Add new data list

4.13.2.2 Add a MashZone list for object links

The procedure of adding a list for object links is identical to the procedure of adding a data list. They only differ in the content. Data lists focus on the data attributes of the object, while a link list provides the IDs of the objects to be linked. It is recommended to use an existing list as a basis.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Generate a view that provides only the IDs and, if required, the names of the objects to be linked. 2. Link the objects within the MashZone Feed Editor.
Documents	mashzone_views.xml, views.xsd
Example	<ul style="list-style-type: none"> ▪ AddModifyMashzoneURL\WEB-INF\config\custom\xml\custom.xml: Add new object relation list ▪ AddModifyMashzoneURL\WEB-INF\config\custom\properties\application\custom.properties: Add new object relation list

4.13.2.3 Assign a name to a MashZone list

You can assign a name to a MashZone list using a property key in a property file. The following naming conventions apply:

- The key must have the format **view.<view ID>.DBI**.
- The file must not contain underscores (_) as separators because the underscore is used as a separator for the country code.

If the name is to be available in different languages, you must generate one file for each language and assign the appropriate country codes. Example: **custom_en.properties** (English) or **custom_de.properties** (German).

Location	Property file in the properties\application folder
Procedure	<ol style="list-style-type: none"> 1. Add a line in a new or an existing file according to the above-mentioned naming convention. 2. Enter a name after the equal sign.
Documents	See Adapt names (Page 3).
Example	AddModifyMashzoneURL \WEB-INF\config\custom\properties\application\custom.properties: Add new data list und Add new object relation list

4.14 Adjust navigation

You can adjust parts of the navigation within the HTML interface of ARIS Risk & Compliance Manager by adapting the navigation XML files. Adjustments can be made for Navigation within the individual interface areas **Home**, **Explorer**, **Evaluation** and **Administration**.

4.14.1 Adapt navigation for an area

The contents of **Home**, **Explorer**, **Evaluation** and **Administration** can be adapted using the associated XML file (navigation_home.xml, navigation_explorer, navigation_evaluation, navigation_administration.xml). For **Home**, the XML file contains information on the menu items to be displayed in the main window. For Explorer and Evaluation, the XML FILE contains information on the structure of the left-hand navigation. For Administration, the XML FILE contains information on the structure of the top navigation bar and the content of each associated tab.

The following elements can be defined within the navigation XML files:

- **<nav.item>**
Used as a structural element for a group of elements. Can alternatively be used as a reference for an element that is defined in the XML files.
- **<nav.data.grid>**
Represented as a link, opens a list. Used in all areas.
- **<nav.evaluation>**
Represented as a link, opens an evaluation. Only used in Evaluation.
- **<nav.job>**
Represented as a link or button, schedules an administrative background job. Only used in Administration.
- **<nav.action>**
Represented as a link or button, immediately performs an action. Only used in Administration.
- **<nav.object>**
Represented as a link, opens an ARIS Risk & Compliance Manager object. Only used in Administration.

You can define the display of these elements via `<nav.access>`. The element is displayed only if all requirements defined in `<nav.access>` are met. The following types of condition exist:

- **<nav.access.component>**
Users must be assigned in ARIS Risk & Compliance Manager to the role that provides them with access to the functions relevant for them.
- **<nav.access.privilege>**
Users must have the specified system privilege for at least one of the roles assigned to them. The privileges are defined in the **roles.xml** file.
- **<nav.access.role>**
Users must have the specified role. The privileges are defined in the **roles.xml** file.

Each type of condition can exist only once in each condition.

A `<nav.item>` element propagates its access conditions to all subordinate elements. If a `<nav.item>` element contains a subordinate `<nav.data.grid>` element, the conditions directly assigned to the elements and those of the superior element are combined.

Example

In the navigation, an existing list is inserted at an additional position and another access privilege is defined for it.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy the <code><nav.module></code> element or individual <code><nav.item></code> elements from the default to the customizing file. 2. Embed them within a <code><navigation></code> element. Subsequently, use the navigation.xsd file to change the contents.
Documents	navigation_explorer.xml, navigation.xsd
Example	Navigation_Module\WEB-INF\config\custom\xml: Modify the module navigation

4.15 Adapt and extend event enabling

The control of processes through events is achieved by processing incoming events that trigger the generation of certain objects in different states in ARIS Risk & Compliance Manager. You cannot adapt this type of control. You can adapt the attributes of the objects generated in ARIS Risk & Compliance Manager, e.g., test cases, by adapting the existing configuration files or by generating an additional configuration file based on existing ones. These adaptations and extensions are carried out by a Complex Event Processing Engine administrator. Two customizing options in ARIS Risk & Compliance Manager are described below. See the Complex Event Processing documentation for information on how the files are to managed in the Complex Event Processing Engine, i.e., which events are sent, which ways are used, and which configuration files they are based on.

4.15.1 Extend existing event type XSDs

The default implementation of the control via events uses configuration files supplied in XSD format for the Complex Event Processing Engine. These files contain all attributes that the relevant event can transfer in ARIS Risk & Compliance Manager. Modifications become effective as soon as the Complex Event Processing Engine administrator saves the changes to the configuration file. If necessary, the configuration file must be updated locally.

Location	XSD file in folder <event architecture installation directory>\common\EventTypeStore\WebM\ARCM
Procedure	Add a new attribute to the configuration file. If you use an ARIS Risk & Compliance Manager attribute you must use the same name as in ARIS Risk & Compliance Manager.
Documents	objectTypes.xml, IncidentEvent.xsd, or TestcaseEvent.xsd

4.15.2 Create new event type XSDs

The default implementation of the control via events uses configuration files supplied in XSD format for the Complex Event Processing Engine. These files contain all attributes that the relevant event can transfer in ARIS Risk & Compliance Manager. You can also generate new configuration files based on the configuration files supplied. However, it is not possible to modify the object type to be generated in ARIS Risk & Compliance Manager. The administrator must not only save the new configuration file, but ensure, that events are sent via a corresponding path based on the new configuration file. The new configuration file can be updated in the local Event Type Store.

Location	XSD file in folder <event architecture installation directory>\common\EventTypeStore\WebM\ARCM
Procedure	Copy the existing XSD file from the ARCM folder to the relevant Event Type Store and add attributes. If you use an ARIS Risk & Compliance Manager attribute you must use the same name as in ARIS Risk & Compliance Manager.
Documents	objectTypes.xml, IncidentEvent.xsd or TestcaseEvent.xsd as a template

4.16 Adapt interface appearance

You can adjust the interface appearance of ARIS Risk & Compliance Manager to provide a customized look and to upgrade custom client-side behavior. The following adjustments can be made:

- Replacing images and icons
- Including additional CSS files to modify the presentation of the HTML interface
- Including additional JavaScript files to modify the client-side behavior

4.16.1 Exchange images and icons

All default images and icons of ARIS Risk & Compliance Manager are stored in the folder `\design\default\images\` and its subfolders. To replace an image or icon, save the new resource in the folder `\design\custom\images\` or its subfolder. Path and file name must match exactly the ones of the replaced image or icon, except the default folder, which is replaced with custom.

Example

To replace the icon **wait.gif** stored in `\design\default\images\icons\`, save the new resource with the name **wait.gif** in `\design\custom\images\icons\`.

This procedure does not work for images and icons declared in default style sheets.

4.16.2 Include CSS files

The design of the HTML interface can be modified by including CSS style sheets. The custom style sheets are inserted after the default style sheets. For the custom style sheet to take effect, the included rule sets must have selectors that are as precise as the default ones. These files are added to all pages.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy the <includesGlobal> element from default configuration to customizing. 2. Create one or more <globalStyle> elements within the <includesGlobal> element. 3. Move style sheets to folder \design\custom\css and images to folder \design\custom\images. 4. Modify attributes of <globalStyle> to match the style sheet's name.
Documents	globalstaticresources.xml, globalstaticresources.xsd
Remark	In some cases, images and icons are embedded in the interface via style sheets. To replace these resources, overwrite the existing CSS rule and include the new image.
Example	AddCustomUIHeader \WEB-INF\config\custom\xml\logo.xml: Change interface

4.16.3 Include JavaScript files

In some cases, the client-side behavior of ARIS Risk & Compliance Manager is modified as part of a customization. For that purpose, JavaScript file can be added to the scope. The custom script files are included after the default ones, meaning that they can override previously defined objects and methods. These files are added to all pages.

Location	XML file in the xml folder
Procedure	<ol style="list-style-type: none"> 1. Copy the <includesGlobal> element from the default configuration to the customizing file. 2. Create one or more <globalScript> elements within the <includesGlobal> element. 3. Move JavaScript files to folder \js\custom\. 4. Modify attributes of <globalScript> to match the file's name.
Documents	globalstaticresources.xml, globalstaticresources.xsd
Example	AddCustomUIHeader \WEB-INF\config\custom\xml\logo.xml: Change interface

5 Disclaimer

ARIS products are intended and developed for use by people. Automatic processes such as generation of content and import of objects/artefacts using interfaces can lead to a huge data volume, processing of which may exceed the available processing capacity and physical limits. Physical limits can be exceeded if the available memory is not sufficient for execution of the operations or storage of the data.

Effective operation of ARIS Risk & Compliance Manager requires a reliable and fast network connection. A network with an insufficient response time reduces system performance and can lead to timeouts.

If ARIS products are used in a virtual environment, sufficient resources must be available to avoid the risk of overbooking.

The system has been tested in the **Internal control system** scenario with 400 users logged in simultaneously. It contains 2,000,000 objects. To guarantee adequate performance, we recommend operating with not more than 500 users logged in simultaneously. Customer-specific adaptations, particularly in lists and filters, have a negative impact on performance.

6 Software AG support

ON THE WEB

With a valid support contract you can access the solution database.

Click <https://empower.softwareag.com/>

For questions about special installations that you cannot carry out yourself, please contact your local Software AG sales organization.

BY PHONE

With a valid support contract you can reach Global Support ARIS at:

+800 ARISHelp

The "+" stands for the respective prefix for making an international connection in this land.

An example of the number to be dialed within Germany using a land line: 00 800 2747 4357