Customizing-Handbuch

ARIS Risk & Compliance Manager
Version 9.8

April 2015

Product Documentation

Dieses Dokument gilt für ARIS Risk & Compliance Manager ab Version 9.8. Hierin enthaltene Beschreibungen unterliegen Änderungen und Ergänzungen, die in nachfolgenden Release Notes oder Neuausgaben bekanntgegeben werden.

Inhalt

1 Textkonventionen

Im Text werden Menüelemente, Dateinamen usw. folgendermaßen kenntlich gemacht:

- Menüelemente, Tastenkombinationen, Dialoge, Dateinamen, Eingaben usw. werden **fett** dargestellt.
- Eingaben, über deren Inhalt Sie entscheiden, werden **<fett und in spitzen Klammern>** dargestellt.
- Einzeilige Beispieltexte werden am Zeilenende durch das Zeichen ↵ getrennt, z. B. ein langer Verzeichnispfad, der aus Platzgründen mehrere Zeilen umfasst.
- Dateiauszüge werden in folgendem Schriftformat dargestellt:

```
Dieser Absatz enthält einen Dateiauszug.
```

2 Was kann angepasst werden?

Die Konfiguration von ARIS Risk & Compliance Manager ist in XML-Dateien definiert, denen kommentierte XML-Schema-Dateien (.xsd) zu Grunde liegen. Diese XML-Dateien beschreiben:

- Die Beschaffenheit von Objekten und Attributen sowie deren Darstellung an der Oberfläche.
- Objektlebenszyklus und Formularfluss
- Hierarchien
- Statistiken und Reporte
- Rollen, Rechte und Aufgabentrennung
- Benachrichtigungen
- Zeitgesteuerte Aufgaben
- Offline-Bearbeitung
- Dashboard-Links

Dazu wird in der XML-Konfiguration verwiesen auf:

- Java-Klassen, die ein bestimmtes Verhalten implementieren
- Property-Dateien mit den lokalisierten Texten für die Oberfläche
- DRL- und DSL-Dateien mit den Regeln für die Formulare

Die Anwendung wird angepasst, indem die XML-Konfiguration angepasst wird, sodass die vorhandenen Elemente neu kombiniert werden, z. B. indem ein vorhandener Attributtyp unter neuem Namen einem bestehenden Objekttyp hinzugefügt wird und neu angelegte Elemente eingebunden werden, z. B. neue Meldungen oder auch Java-Klassen, die neues Verhalten implementieren, das sich nicht per XML konfigurieren lässt. Dazu ist kein separater Bauprozess notwendig. Änderungen an der Konfiguration werden bei laufendem Server aktualisiert, sofern sich das System im Testmodus befindet. Bei Produktivsystemen ist ein Neustart des Servers erforderlich. Im Abschnitt **Elementare Anwendungsfälle (Seite 22)** sind alle Schritte beschrieben, aus denen sich die möglichen Anpassungen, ausgehend von der XML-Konfiguration, zusammensetzen.

An einigen Stellen kann das Verhalten zusätzlich angepasst werden, indem per XML-Konfiguration eigene Java-Implementierungen spezieller Interfaces eingebunden werden. Diese Java-Klassen können gegen ein definiertes Interface unabhängig entwickelt und dann wie die Standard-Klassen eingebunden werden, ohne dass ein spezieller Bauprozess für ARIS Risk & Compliance Manager notwendig ist.

Falls es eine aktualisierte Version dieses Dokuments gibt, finden Sie diese hier: http://aris.softwareag.com/ARISDownloadCenter/ADCDocumentationServer (http://aris.softwareag.com/ARISDownloadCenter/ADCDocumentationServer)

3 Allgemeines Vorgehen

Die Standardimplementierung basiert auf Konfigurationsmechanismen, die auch für Anpassungen benutzt werden. Die Standardkonfiguration des kontrollbasierten Ansatzes (CBA) ist bereits eine Anpassung der Standardkonfiguration des risikobasierten Ansatzes (RBA). Soll das Standardverhalten oder die Standardstruktur geändert werden, können die entsprechenden Bereiche in der XML-Konfiguration überschrieben werden, sodass an dieser Stelle punktuell ein verändertes Verhalten bzw. eine veränderte oder erweiterte Struktur definiert wird. Diese punktuellen Änderungen und Erweiterungen werden unterhalb des Ordners **tomcat\webapps\arcm\WEB-INF\config\custom**, der sich im Installationsordner befindet, vorgenommen.

Im Kapitel **Elementare Anwendungsfälle** (Seite 22) sind die nötigen Schritte für angepasste Konfigurationen anhand der unterstützten Anwendungsfälle beschrieben. Die dort beschriebene Vorgehensweise ist in Konfigurationsbeispielen umgesetzt, die sich im Ordner **customizing examples** befinden.

Im Ordner **standard configuration** befinden sich zusätzlich die Standard-XML-Konfigurationsdateien. Diese können als Ausgangspunkt von Anpassungen dienen, indem die anzupassenden Passagen von dort kopiert und gemäß den individuellen Anforderungen verändert werden.

3.1 Anpassen der XML-Konfiguration

Der Ordner **xml** enthält eine oder mehrere XML-Dateien mit der angepassten XML-Konfiguration. Diese werden vom System gegen die XML-Schema-Datei **custom.xsd.** im Ordner **xsd** validiert, die selbst nicht verändert werden darf. Das Wurzel Element dieser XML-Dateien muss das Element **<custom>** sein.

3.2 Regeln anpassen

Der Ordner **rules** enthält angepasste Regeldateien, die von der XML-Konfiguration eingebunden werden. Siehe **Regel hinzufügen/anpassen** (Seite 77).

3.3 Namen anpassen

Der Ordner **properties** enthält eine oder mehrere Property-Dateien mit angepassten Strings für die Oberfläche.

Namenskonventionen:

- Der Dateiname muss mit **_xx.properties.** enden. (**xx** steht für das Kürzel der Sprache, in der die Strings lokalisiert sind.)
- Der Unterstrich (_) darf an keiner anderen Stelle des Dateinamens vorkommen.

Beispiel

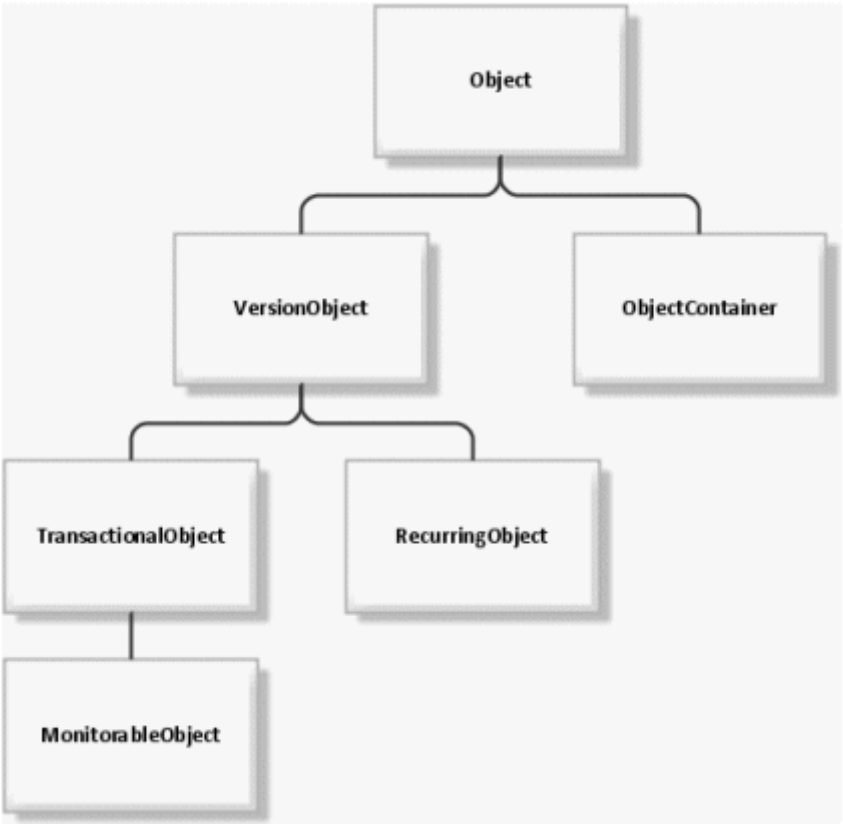
Die englische Sprachversion könnte **myCustomizedStrings_en.properties heißen, aber nicht my_customized_strings_en.properties.**

Auf diese Weise lassen sich verschiedene Sprachversionen parallel anlegen, deren Dateinamen sich nur durch das Sprachkürzel unterscheiden.

3.4 Vererbung

Mit der Version 4.0 von ARIS Risk & Compliance Manager steht in der Objektkonfiguration (**objectTypes.xml**) ein Vererbungsmechanismus zur Verfügung. Damit wird eine einheitliche Objekt- und Attributstruktur der Komponenten, Test-Management, Issue-Management usw., gewährleistet, sowie Objekte mit ähnlicher Bedeutung und Funktion. Durch die Vererbung wird vorgegeben, welche Funktion ein Objekttyp innerhalb der Komponente hat und dadurch der Konfigurationsaufwand verringert. Letzteres geschieht durch die Zentralisierung und Wiederverwendung Workflow-relevanter Attribute. Ein weiterer Vorteil des Vererbungsmechanismus ist, dass die Programmierung generischer Systemfunktionen, z. B. Monitoring, erleichtert wird, da auf die zentral konfigurierten Attribute zugegriffen werden kann.

3.4.1 Übersicht



3.4.2 Objekttypen Object und VersionObject

Die beiden Objekttypen **Object** und **VersionObject** (objectTypes.xml: **OBJECT, VERSION**) enthalten zentrale technische Attribute. Sie sollten im Rahmen eines Customizings nicht verändert werden. Objekttypen, die dem Versionierungsmechanismus von ARIS Risk & Compliance Manager unterliegen sollen, müssen den Objekttyp **versionObject** erweitern. Unversionierte Objekttypen erben direkt von **Object**. Das Erweitern des Objekttyps **Object** muss nicht explizit angegeben werden, ähnlich wie bei der Programmiersprache Java, sondern erfolgt automatisch.

3.4.3 Objekttyp TransactionalObject

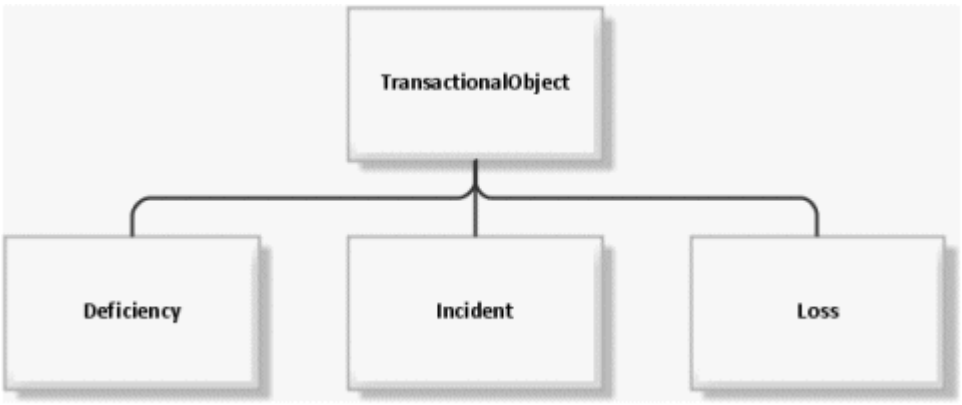
Der Objekttyp **transactionalObject** (objectTypes.xml: **TRANSACTIONAL**) fasst einige Attribute zusammen, die zu den typischen Bewegungsdatenobjekten gehören. In der Regel sind das die Objekttypen, z. B. Testfall, die im Rahmen eines Workflows verschiedene Rollen, z. B. Tester und Test-Reviewer, durchlaufen und die Basis der in ARIS Risk & Compliance Manager erfassten Daten bilden.

Attribute

Attribut-ID	Datentyp	Verwendungszweck
owner_status	Aufzählung	Status
owner_group	Zuweisung	Verantwortliche Gruppe für die Ausführung
owner	Zuweisung	Ausführender Benutzer
owner_substitute	Zuweisung	Vertretung des ausführenden Benutzers
execution_date	Datum	Ausführungsdatum
reviewer_status	Aufzählung	Review-Status
reviewer_group	Zuweisung	Verantwortliche Gruppe für den Review
reviewer	Zuweisung	Review-Benutzer
reviewer_substitute	Zuweisung	Vertretung des Review-Benutzers
review_date	Datum	Review-Datum

Werden für verschiedene Workflows unterschiedliche Aufzählungen für die beiden Status-Attribute benötigt, können diese am eigentlichen (erbenden) Objekttyp überschrieben werden. Bei den Gruppenzuweisungsattributen muss die Auswahl am erbenden Objekttyp auf eine bestimmte Rolle beschränkt werden.

Vererbungsdiagramm



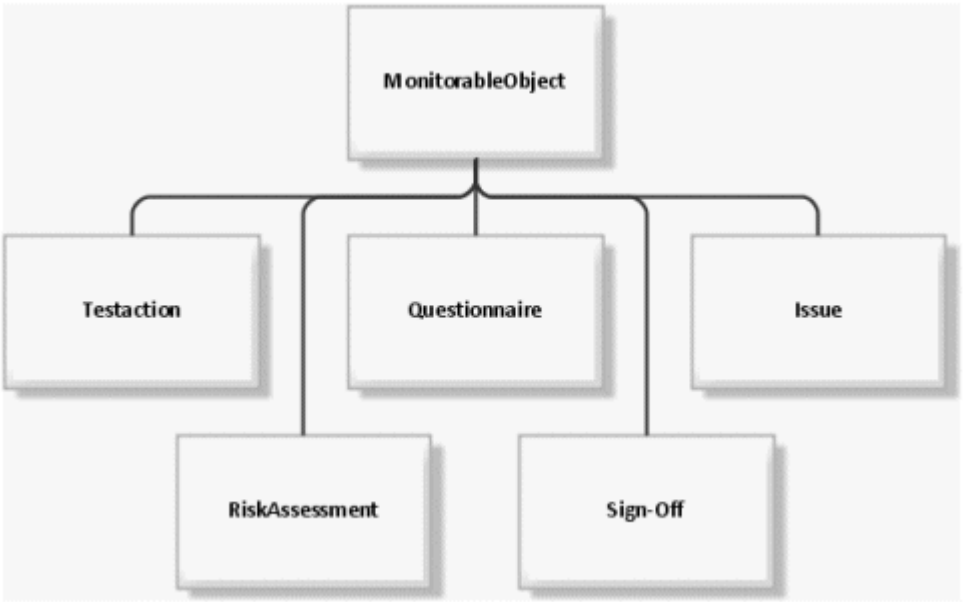
3.4.4 Objekttyp MonitorableObject

Der Objekttyp **monitorableObject** (objectTypes.xml: **MONITORABLE**) erweitert den zuvor beschriebenen Objekttyp **transactionalObject** um einige Attribute, die das zeitbasierte Monitoring betreffen. Bewegungsdatentypen, die ein Ablaufdatum haben, z. B. Testfall, das von der Anwendung überwacht wird, sollten von diesem Objekttyp erben.

Attribute

Attribut-ID	Datentyp	Verwendungszweck
plannedstartdate	Aufzählung	Startdatum des Bearbeitungszeitraumes
plannedenddate	Zuweisung	Enddatum des Bearbeitungszeitraumes
controlstartdate	Zuweisung	Startdatum des Kontrollzeitraums
controlenddate	Zuweisung	Enddatum des Kontrollzeitraums

Vererbungsdiagramm



3.4.5 RecurringObject

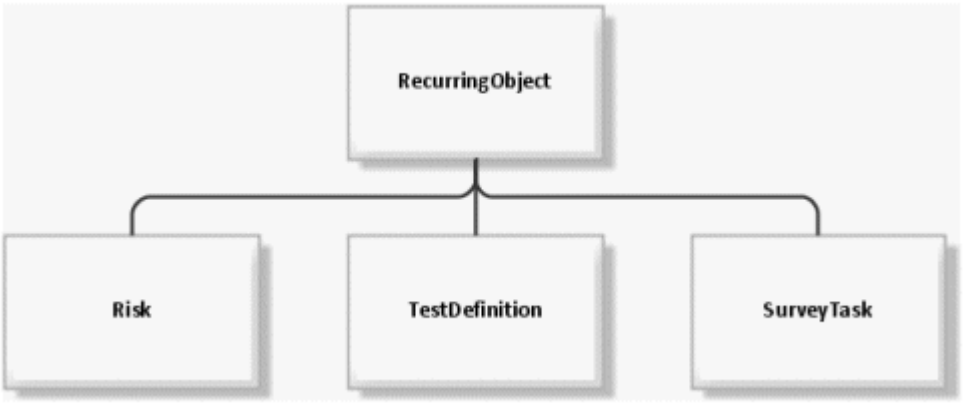
Der Objekttyp **RecurringObject** (objectTypes.xml: **RECURRING**) gehört zu den Stammdaten. Er fasst Attribute zusammen, die zum erneuten Generieren von Bewegungsdatenobjekten benötigt werden.

Attribute

Attribut-ID	Datentyp	Verwendungszweck
owner_group	Zuweisung	Verantwortliche Gruppe für die Ausführung
frequency	Aufzählung	Frequenz, mit der die Bewegungsdatenobjekte generiert werden (einmalig, täglich, wöchentlich usw.)
duration	Integer (long)	Frist zur Ausführung in Tagen
startdate	Datum	Datum, ab dem die Bewegungsdaten regelmäßig generiert werden
enddate	Datum	Datum, bis zu dem die Bewegungsdaten regelmäßig generiert werden
control_period	Aufzählung	Länge des Kontrollzeitraums (Tag, Woche, Monat usw.)
offset	Integer (long)	Offset des Kontrollzeitraums in Tagen
reviewer_group	Zuweisung	Verantwortliche Gruppe für den Review

Werden für verschiedene Workflows unterschiedliche Aufzählungen für **frequency** und **control_period** benötigt, können diese am eigentlichen (erbenden) Objekttyp überschrieben werden. Bei den Gruppenzuweisungsattributen muss die Auswahl am erbenden Objekttyp auf eine bestimmte Rolle beschränkt werden. Deshalb muss dieses Attribut überschrieben und mit der korrekten Rolleneinschränkung versehen werden.

Vererbungsdiagramm



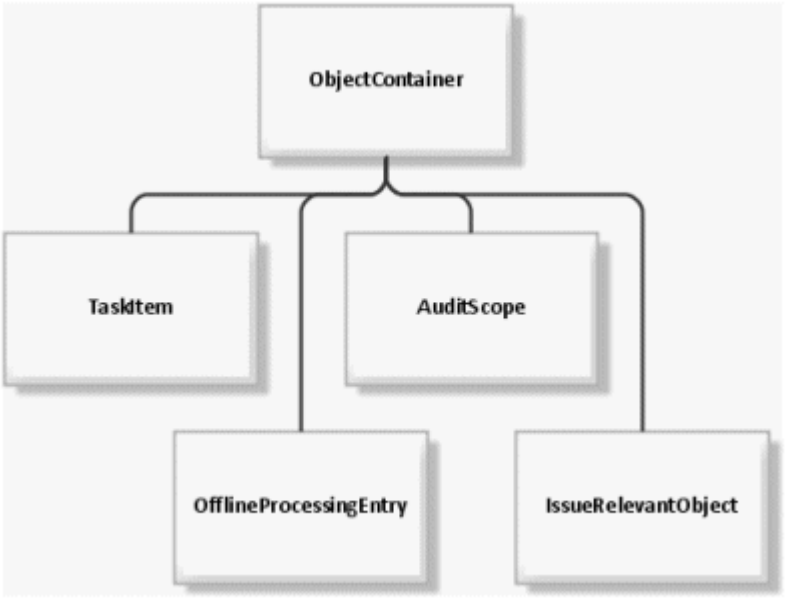
3.4.6 ObjectContainer

Der Objekttyp **objectContainer** (objectTypes.xml: **OBJECTCONTAINER**) dient als Container für andere Objekte. Anwendung findet er z. B. beim Issue-Management, wo Objekte beliebigen Typs als Issue-relevante Objekte verknüpft werden können.

Attribute

Attribut-ID	Datentyp	Verwendungszweck
object_id	Ganzzahl (long)	ID des beinhalteten Objekts
object_version_number	Ganzzahl (long)	Versionsnummer des beinhalteten Objekts
object_objtype	String	Objekttyp des beinhalteten Objekts
object_clientSign	String	Hilfsattribut für den Mandantenfilter
object_clientSigns	String	Hilfsattribut für den Mandantenfilter (enthält eine Liste der zugewiesenen Mandantenkennungen, die durch Kommas getrennt sind)
object_name	String	Name des beinhalteten Objekts
object_ovid	String	Hilfsattribut für die Selektion (Object-Version-ID)
role	Aufzählung	Rolle, mit der auf das beinhaltete Objekt zugegriffen wird bzw. wurde

Vererbungsdiagramm



3.4.7 Vererbung in der Datei objectTypes.xml

Die Vererbung wird in der Datei **objectTypes.xml** durch das XML-Attribut **extends** am XML-Element **objectType** ausgedrückt. Der Wert des Attributs muss die ID des übergeordneten Objekts enthalten.

Basis-Objekte mit fester Bedeutung

OBJECT, VERSION, TRANSACTIONAL, RECURRING, MONITORABLE, OBJECTCONTAINER

Vererbungsstruktur

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

CLIENT > 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 Konventionen

3.5.1 Konventionen in der XML-Konfiguration

Mit der Version 4.0 von ARIS Risk & Compliance Manager werden viele Konventionen eingeführt, die den Konfigurationsaufwand verringern. Beispiele für solche Konventionen sind Lokalisierungsschlüssel (Property-Keys), die sich aus Objekt- und Attributnamen ergeben sowie Schaltflächen, die Namenskonventionen unterliegen. In solchen Fällen werden die Schlüssel und Dateinamen anhand der Namenskonventionen ermittelt und die entsprechenden Ressourcen automatisch geladen. Die Funktion dieser Konventionen, ist zu jeder XML-Datei im zugehörigen Schema (XSD) dokumentiert.

3.5.2 Konventionen bei der Objektgenerierung

Bei der Objektgenerierung werden ebenfalls Konventionen genutzt, um den Customizing-Aufwand zu reduzieren. Ein wesentliches Beispiel ist der automatische Transport von Attributen von Stammdaten zu Bewegungsdaten. So wird beispielsweise bei der Testfallgenerierung das Attribut **test activities** (testingsteps) anhand einer Namenskonvention automatisch von der Testdefinition zum generierten Testfall transportiert. Wurde in früheren Versionen von ARIS Risk & Compliance Manager ein neues Attribut eingeführt, das ebenfalls zu einem Bewegungsdatenobjekt transportiert werden sollte, musste neben der XML-Konfiguration zusätzlich der entsprechende Objektgenerator im Java-Quellcode angepasst werden. Ab Version 4.0 genügt es, die Attribute an Quell- und Zielobjekt identisch zu benennen.

3.5.2.1 Mandantenzugehörigkeit bei mandantenspezifischen Objekten

Bei mandantenspezifischen Objekten wird die Mandantenzugehörigkeit des Quellobjekts automatisch am Zielobjekt übernommen.

3.5.2.2 Objekttyp MonitorableObject

Bei der Generierung von Objekten des Typs **monitorableObject** werden Attribute vom entsprechenden **recurringObject** übernommen und anhand der Stammdatenattribute berechnet, z. B. Start- und Enddatum. Voraussetzung dafür ist, dass im Kontext ein entsprechendes **recurringObject** gefunden wird. In der Standardkonfiguration von ARIS Risk & Compliance Manager gibt es folgende Beziehungen zwischen Recurring- und Monitoring-Objekten:

TESTDEFINITION > TESTCASE

SURVEYTASK > SURVEY

SURVEYTASK > QUESTIONNAIRE

RISK > RISKASSESSMENT

Im Einzelnen ist die Behandlung der Attribute in folgender Tabelle dargestellt:

Attribut	**Behandlung**
plannedstartdate	Wird aus dem Startdatum (**startdate**) und der Frequenz (**frequency**) des Quellobjekts berechnet.
plannedenddate	Wird aus dem Startdatum des Zielobjekts (**plannedstartdate**) und der Dauer des Quellobjektes (**duration**) berechnet.
controlenddate	Wird aus dem Startdatum des Zielobjekts (**plannedstartdate**) und dem Offset des Quellobjekts (**offset**) berechnet.
controlstartdate	Wird aus dem Ende des Kontrollzeitraums (**controlenddate**) und des Kontrollzeitraums (**control_period**) des Quellobjekts berechnet.
owner_group	Wird direkt vom Quellobjekt übernommen.
reviewer_group	Wird direkt vom Quellobjekt übernommen.

3.5.2.3 Gleichheit bei Attributnamen

Attribute des Zielobjekts, zu denen es identisch benannte Attribute an einem der Quellobjekte gibt, werden automatisch übernommen. Das gilt allerdings nicht für Attribute, die von einem der Basisobjekttypen **Object** (**OBJECT**) oder **versionObject** (**VERSION**) geerbt wurden. Es kann sinnvoll sein, Attribute identisch zu benennen, die automatische Übernahme der Werte aber zu unterdrücken. Dazu kann der Hilfsklasse, in der die Konventionen umgesetzt sind, eine Liste mit Attributnamen übergeben werden, die übersprungen werden sollen.

3.5.2.4 Objektzuweisung bei Namensgleichheit

Zielobjekte, bei denen es sich um Bewegungsdaten handelt, werden oft mit Quellobjekten als Attribut verlinkt, damit bei der Bearbeitung des Zielobjekts durch den Endbenutzer weitere Informationen zur Verfügung stehen, die zum Verständnis der Aufgabe benötigt werden. Die Verlinkung der Quellobjekte geschieht ab der Version 4.0 von ARIS Risk & Compliance Manager automatisch, wenn der Name des Attributs am Zielobjekt und der Name des Objekttyps eines Quellobjekts übereinstimmen. Beispielsweise wird beim Generieren eines Testfalls das Attribut **Risiko** mit dem entsprechenden Risiko (**RISK**) belegt, das als Quellobjekt übergeben wurde.

3.6 Class-Mappings

Ein Class-Mapping verbindet eine bestimmte Implementierung (Klasse) mit einem Namen. Diese Namen werden in anderen Teilen der Konfiguration von ARIS Risk & Compliance Manager verwendet, um auf die gewünschte Implementierung zu verweisen. Der Name ist dabei deutlich kürzer und einprägsamer als der lange Name der Klasse. Somit wird die Übersichtlichkeit in der Konfiguration gewahrt. Innerhalb von ARIS Risk & Compliance Manager gibt es mehrere verschiedene Class-Mappings. Im Folgenden werden ihre Definition, Verwendung und Einsatzbereiche beschrieben.

3.6.1 Actions

Action-Commands werden in der Benutzeroberfläche eingesetzt, um die Interaktionen von Benutzern in Befehle der Business-Logik umzusetzen. Hierfür genügt in den meisten Fällen die Standardimplementierung. In einigen Sonderfällen oder im Customizing muss das Verhalten angepasst oder ergänzt werden. Dazu gibt es ein Class-Mapping, welches die Anpassung vereinfacht. Alle Klassen, die diesem Mapping zugeordnet werden, müssen das Interface **IActionCommand** enthalten.

Die Definition der Action command mappings besteht aus mehreren Teilen. Zum einen die Definition der ActionCommandIds, die eine Liste aller gültigen Befehle und deren Erläuterung enthält.

```
<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>
```

Weiterhin gibt es die folgenden Bereiche:

- **objectTypeCommands**
  Definieren Befehle, welche sich auf ein oder mehrere Objekte auswirken. Sie werden meist auf Formularen verwendet.
- **listCommands**
  Definieren Befehle, welche die Listen steuern, z. B. Paging, Filter anwenden usw.

- **evaluationCommands**
  Definieren Befehle, welche die Auswertungen steuern, z. B. die Baumstruktur aufklappen, Filter anwenden usw.
- **jobCommands**
  Definieren Befehle, welche auf die verschiedenen Eigenheiten der Jobs eingehen und diese entsprechend ausführen.
- **dialogCommands**
  Definieren Befehle, welche die Dialoge steuern.

Jeder dieser Bereiche besteht aus einer Liste von <commandSet>-Elementen. Jedes Set verlangt ein name-Attribut. Dieses gibt an, worauf sich die folgenden Command-Definitionen beziehen. Bei objectTypeCommands ist dies der eindeutige Identifizierer des Objekttyps, bei listCommands der eindeutige Identifizierer der Liste usw.

Ein besonderes commandSet ist common. Hier werden die Standardimplementierung für alle Objekttypen, Listen usw. hinterlegt, sodass nur die speziellen Implementierungen gesondert angegeben werden müssen. Verwenden beispielsweise mehrere Listen die gleiche Implementierung, können alle Listen durch Kommas getrennt als Name des commandSet eingetragen werden. Eine Liste kann gleichzeitig in mehreren commandSets enthalten sein.

```
<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

Sektion **Commands**

Commands werden im Zusammenhang mit den Workflows verwendet, um die Business-Logik auszuführen. Commands sind meist sehr kompakt gehalten und auf eine Aufgabe spezialisiert, um wiederverwendbar zu sein. Sie können mit Parametern versehen werden, um für den jeweiligen Einsatzzweck (Command-Chain) Anwendung zu finden. Daneben gibt es auch eine ganze Reihe von Spezialimplementierungen, die genau für einen bestimmten Einsatzzweck geschrieben sind und sich nicht oder kaum wiederverwenden lassen. Sie können aber als Vorlage für eigene Commands herangezogen werden. Commands müssen das Interface **ICommand** implementieren.

3.6.3 Statistic-Class-Mappings

In diesem Class-Mapping befinden sich alle Klassen mit Alias-Namen, die in den Statistiken verwendet werden. Hinweise zur Verwendung dieser Klassen befinden sich im Kapitel **Statistiken hinzufügen / anpassen** (Seite 62). Details zur Implementierung weiterer Klassen sind jeweils im Java-Doc des zu implementierenden Interface der Sektion beschrieben.

Sektion evaluationAccessControl

Access-Control-Implementierungen werden benötigt, um nur bestimmten Benutzern Zugriff auf bestimmte Statistiken zu gewähren. Diese Klassen implementieren das Interface **IEvaluationAccessControl**.

Sektion statisticTreeProvider

Tree-Provider-Implementierungen werden benötigt, um die Hierarchiestrukturen, auf welchen die Statistiken aufbauen (also die Struktur in der ersten Spalte), zu generieren.

Sektion statisticDataFilter

Data-Filter-Implementierungen werden verwendet, um die Daten für eine Statistik zu filtern. Diese Klassen implementieren das Interface **IStatisticDataFilter**.

Sektion statisticDataSource

Data-Source-Implementierungen werden verwendet, um die Datenquellen einer Statistik zu konfigurieren. Im Standard gibt es nur die Datenquellen **view** und **tree**. **view** erlaubt den direkten Zugriff auf die Datenbank von ARIS Risk & Compliance Manager, **tree** wird verwendet, um den Tree-Provider als Datenquelle zu verwenden. Diese Klassen implementieren das Interface **IStatisticDataSource**.

Sektion statisticCalculator

Calculator-Implementierungen werden zur Aufbereitung der anzuzeigenden Daten verwendet. Sie wandeln die technischen Daten, welche von Data-Source-Implementierungen zur Verfügung gestellt werden in von Benutzern lesbare Daten um. Diese Klassen implementieren das Interface **IStatisticCalculator**.

Sektion statisticDataLinker

Data-Linker-Implementierungen werden zum Verlinken der anzuzeigenden Daten verwendet. Eine Verlinkung kann z. B. eine Detailansicht der Daten als Liste oder eine weitere Statistik sein. Diese Klassen implementieren das Interface **IStatisticDataLinking**.

3.6.4 BI-Class-Mappings

Sektion **PredefinedValueProvider**

Ein Value-Provider erlaubt es, aus dem Kontext des Benutzers und evtl. weiteren Parametern die Optionen einer Auswahl im Dialog automatisch zu generieren. Diese Auswahlboxen kommen sehr häufig vor und haben immer wieder ähnlichen, aber benutzerspezifischen Inhalt. Hierfür gibt es einige Standardimplementierungen, die viele Fälle abdecken.

3.6.5 Ui-Class-Mappings

Hier finden sich die Mappings, welche die Benutzeroberfläche gestalten.

Sektion Renderer

Ein Renderer generiert ein HTML-Fragment, um ein Attribut in Formularen oder Listen darzustellen. Ein Renderer muss das Interface **IRenderer** implementieren. Neben der Klasse für die HTML-Darstellung (Attribut **reportClsName**) kann auch eine Klasse angeben werden, um die Darstellung in den PDF-/Excel-Reporten anzupassen (Attribut **clsName**).

Sektion FilterRenderer

Filter-Renderer unterscheiden sich von den normalen Renderern lediglich in der Implementierung. Hier wird auf einige Sonderfälle Rücksicht genommen, die die Darstellung der Filter betreffen.

Sektion ColumnRenderer

Column-Renderer implementieren das Interface **IColumnRenderer** und werden von den konfigurierbaren Statistiken verwendet, um die Datenzellen darzustellen.

Sektion Layouter

Layouter generieren HTML-Fragmente, in dem sie ein oder mehrere Renderer passend für das jeweilige **Control** kombinieren. Layouter implementieren das Interface **ILayouter**.

Sektion Controls

Controls, z. B. ein Formular oder eine Liste, kombinieren die HTML-Fragmente der Layouter und fügen weitere Elemente wie Schaltflächen hinzu, um eine vollständig interaktive HTML-Seite zu generieren. Ein Control kann aus mehreren Komponenten bestehen. Falls Teile eines Controls oder seiner Komponenten überschrieben werden sollen, kann ein Control auch auf ein anderes Control verweisen (Attribut **extends**), dann müssen die zu modifizierenden Komponenten neu definiert werden.

```
<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

Um die Menge der Daten einer konfigurierten Ansicht zusätzlich anpassen zu können, gibt es die Möglichkeit, mit Hilfe des Attributs **viewHandler** an einem Element **<view>** eine zusätzliche Handler-Klasse anzugeben. Konvention für diese Handler-Klasse ist, dass sie das Interface **IViewHandler** implementiert. Innerhalb dieser Klasse kann dann die Menge der zurückgelieferten Daten durch zusätzliche Angaben angepasst werden. Diese Form des Customizings ist dann erforderlich, wenn sich die zusätzlich benötigten Anpassungen nicht vollständig konfigurieren lassen.

3.6.7 Konfigurationsdatei VCREG.XML

Die Konfigurationsdatei **vcreg.xml** registriert Validierer und Konvertierer zur Verwendung an den Attributen der Objektdefinitionen von ARIS Risk & Compliance Manager (siehe **Validator zuweisen** (Seite 24)). Dabei wird ein Name des Validierers oder Konvertierers definiert, der auf einen voll qualifizierten Klassennamen verweist.

Beispiel

```
<validator name="minlength"

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

In diesem Beispiel wird ein Validierer **minlength** definiert, dessen Implementierung im Attribut **clsName** angeführt ist. Bei den Validierern wird zusätzlich im Attribut **propertyKey** eine Property angeführt, die im Falle einer negativen Validierung auf der Oberfläche angezeigt wird.

3.7 Hilfe anpassen

Die Hilfe kann durch neu angelegte HTML-Seiten erweitert oder durch Verlinkung auf bestehende Seiten angepasst werden. Außerdem können bestehende Seiten inhaltlich angepasst werden.

Speicherort	Property-Datei im Ordner **properties/help**
Vorgehen	Kopieren Sie die neu zu verwendende HTML-Seite für die Hilfe zunächst nach **webapps\arcm\help\<Sprachkürzel>\embedded**. Der Name der Datei muss so lauten wie die entsprechende Help-ID. Nachdem die Seite in den Hilfeordner kopiert wurde, tragen Sie in der oben angegebenen Property-Datei den neuen Property-Key in eine neue Zeile ein. Diesem neuen Property-Key weisen Sie die neue Help-ID als Wert zu. Für alle Sprachen gibt es eine gemeinsame Property-Datei.
Bemerkung	Laut Konvention wird automatisch beim Anlegen eines neues Formulars, einer neuen Liste oder einer neuen Statistik eine entsprechende Hilfeseite erwartet. Sollte an dieser Stelle generell keine Hilfeseite benötigt werden, muss der laut Konvention vorgegebene Property-Key in der Property-Datei eingetragen werden, ohne einen Wert zuzuweisen. In einem solchen Fall wird keine Hilfeschaltfläche angezeigt. Wollen Sie keine neue Hilfeseite anlegen, kann auch auf eine bestehende Hilfeseite verwiesen werden. Die Verlinkung erfolgt dabei wie oben beschrieben durch die Zuweisung der bestehenden Help-ID in einer der dafür vorgesehenen Property-Dateien. Konventionen für die Benennung neuer Property-Keys: Formular: ARCM_FORM_**[FORM ID]**_PAGE_**[PAGE ID]**.HLP Liste: ARCM_LIST_**[LIST ID]**.HLP Statistik: ARCM_EVALUATION_**[EVALUATION ID]**.HLP

4 Elementare Anwendungsfälle

4.1 Objekteigenschaften anpassen

4.1.1 Überschreiben der Schemaversion

Sobald Objekteigenschaften angepasst werden, ist es zwingend erforderlich, das Schema-Tag aus der Datei **objectTypes.xml** im Customizing zu überschreiben. Nur so kann gewährleistet werden, dass Datenexporte und -importe aus der angepassten Version von ARIS Risk & Compliance Manager eindeutig zugeordnet werden können. Das Anpassen der Version dient außerdem als fester Ausgangspunkt für zukünftige Migrationen. Wenn das Schema-Tag nicht überschrieben wird, Sie aber dennoch Änderungen am Schema vornehmen, kann der Server von ARIS Risk & Compliance Manager nicht gestartet werden.

Vorgehen

Tragen Sie den Namen des Kundenprojekts ohne Leerzeichen in das Customizing-Attribut im überschriebenen Schema-Tag ein. Dieses Attribut hat den Wert **standard** wenn das Schema nicht angepasst wurde.

Beispiel

Eintrag für ein Kundenprojekt mit dem Namen **United Motor Group** auf Basis der -Version 4.0.0.2 von ARIS Risk & Compliance Manager:

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

Sollen verschiedene Kundenversionen basierend auf einer einzigen Version von ARIS Risk & Compliance Manager ausgeliefert werden, können Sie dies durch eine Versionsangabe im Projektnamen kennzeichnen.

Beispiel

Eintrag für ein Kundenprojekt mit dem Namen **United Motor Group Version 1** auf Basis der Version 4.0.0.2 von ARIS Risk & Compliance Manager:

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

Sind die Änderungen im Schema von ARIS Risk & Compliance Manager bedingt durch Änderungen im Datenimport aus ARIS Architect, muss das Zielschema auch in der benötigten Mapping-Datei **Aris2arcm-mapping_[APPROACH].xml** angepasst werden.

Vorgehen

1. Suchen Sie den infoHeader-Tag mit dem Attribut **standard**:
   schema_version="arcm_4.0.0.2_rba_standard"
2. Ersetzen Sie standard mit dem Namen des Kundenprojekts.

Beispiel

schema_version="arcm_4.0.0.2_rba_UnitedMotorGroup "

4.1.2 Einfaches Attribut hinzufügen / ändern

4.1.2.1 Einfaches Attribut anlegen

4.1.2.1.1 Objekttyp anpassen

Wenn Sie einen Objekttyp anpassen wollen, müssen Sie zunächst das Original in die Customizing-Datei kopieren. Dann können Sie die Eigenschaften und Attribute des Objekttyps ändern.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Kopieren Sie das Element **<objectType>** aus dem Standard in das Customizing. Legen Sie dann neue Attribute innerhalb des Elements **<objectType>** an. Sie müssen mindestens die Eigenschaft **id** setzen. Der Wert muss dabei innerhalb des Objekttyps eindeutig sein.
Dokumente	objectTypes.xml, objectTypes.xsd
Beispiel	ModifyObject_AddSimpleAttribute \WEB-INF\config\custom\xml\custom.xml: Add Simple Attribute

4.1.2.1.2 Propertys hinzufügen / anpassen

Properties werden für die Mehrsprachigkeit der Anwendung verwendet. Für jede Sprache gibt es eine eigene Datei. Diese Dateien enthalten das Länderkürzel als Namens-Suffix.

Speicherort	Property-Datei im Ordner **properties / application**
Vorgehen	Tragen Sie den neuen Property-Key gefolgt von einem Gleichheitszeichen und der entsprechenden Übersetzung in eine neue Zeile ein. Für jede Sprache gibt es eine eigene Datei.
Dokumente	Siehe Anpassen der Bezeichnungen.
Beispiel	ModifyObject_AddSimpleAttribute \WEB-INF\config\custom\properties\application\custom.properties: Add Simple Attribute

4.1.2.1.3 Validator zuweisen

Um zu garantieren, dass nur bestimmte bzw. erlaubte Werte in die Datenbank gelangen, kann einem Attribut kann ein Validator zugewiesen werden,

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Fügen Sie ein neues Element **<validate>** als Unterelement des Attributs hinzu und spezifizieren Sie als Name einen registrierten Validator.
Dokumente	vcreg.xml, vcreg.xsd
Beispiel	ModifyObject_AddSimpleAttribute \WEB-INF\config\custom\xml\custom.xml: Add Simple Attribute

4.1.2.1.4 Konverter zuweisen

Sie müssen dem Attribut ggf. einen Konverter zuweisen, der die Daten zwischen der Anwendung und der Datenbank modifiziert. Insbesondere die Konverter **startdate** und **enddate** werden häufig verwendet, wenn zwei Datumsfelder einen Zeitraum beschreiben sollen.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Fügen Sie ein neues Element **<convert>** als Unterelement des Attributs hinzu und spezifizieren Sie als Namen einen registrierten Konverter.
Dokumente	vcreg.xml, vcreg.xsd
Beispiel	Siehe **Validator zuweisen** (Seite 24).

4.1.2.2 Einem Formular ein Attribut hinzufügen

4.1.2.2.1 Formular anpassen

Damit Attribute an der Benutzeroberfläche angezeigt werden und Sie diese bearbeiten können, müssen sie im Formular des Objekttyps spezifiziert werden. Da die Reihenfolge der Attribute wichtig sein kann, wird diese getrennt von der Objekttypdefinition spezifiziert.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	1. Kopieren Sie das Element **<form>** aus dem Standard in das Customizing und fügen Sie ein neues Element **<row>** an der Stelle ein, an der das Attribut erscheinen soll. 2. Legen Sie einen Unterknoten **<element>** mit der Eigenschaft **attribute.idref** an. Hier muss der eindeutige Name des Attributs angegeben werden.
Dokumente	forms_[module].xml, forms.xsd
Beispiel	ModifyObject_AddSimpleAttribute \WEB-INF\config\custom\xml\custom.xml: Modify Form

4.1.2.2.2 Properties für Formular hinzufügen / anpassen

Siehe **Properties hinzufügen / anpassen** (Seite 36).

Vorgehen	Legen Sie am Element **<row>** die Eigenschaft **PropertyKey** fest.
Bemerkung	Dies ist im Formular optional und nur notwendig, falls der Property-Key des ersten Unterknotens **<element>**, welcher per Konvention verwendet wird, keine zutreffende Beschreibung liefert. Steht beispielsweise ein Start- und ein Enddatum in einer Zeile, sollte ein neuer Property-Key definiert werden, der den Namen des entsprechenden Zeitraums wiederspiegelt.
Beispiel	ModifyObject_AddSimpleAttribute \WEB-INF\config\custom\properties\application\custom.properties: Modify Form

4.1.2.2.3 Renderer zuweisen

Für alle Attribute ist eine Standdarddarstellung definiert. Um sie zu verändern, müssen Sie eine andere Klasse spezifizieren, welche diese Darstellung generiert oder mittels Standard-Renderer über Parameter die Darstellung anpassen.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Ordnen Sie der Eigenschaft **template** von **<element>** einen gültigen und registrierten Renderer zu.
Dokumente	uiClassMappings.xml, uiClassMappings.xsd
Beispiel	ModifyObject_AddSimpleAttribute \WEB-INF\config\custom\xml\custom.xml: Assign Renderer

4.1.2.2.4 Regeln anpassen

Standardmäßig sind neue Attribute optional, sichtbar und nicht bearbeitbar. Um dies zu ändern, müssen Sie die Regeln anpassen. In den meisten Fällen gibt es schon Regeln, die die Sichtbarkeit und Bearbeitbarkeit von Attributen bestimmen. Stimmen die Vorbedingungen der Regel, kann das neue Attribut einfach hinzugefügt werden. Werden andere Voraussetzungen gebraucht, muss eine neue Regel angelegt werden.

Speicherort	DRL-Datei im Ordner **rules**.
Vorgehen	Kopieren Sie die Standard-Datei in das Customizing. Anschließend können Sie die Regeln modifizieren oder neue hinzufügen.
Dokumente	Siehe Kapitel **Regel hinzufügen / anpassen** (Seite 77).
Beispiel	ModifyObject_AddSimpleAttribute \WEB-INF\config\custom\rules\usergroup.drl: Modify Form

4.1.2.2.5 Reporte hinzufügen / anpassen

Normalerweise werden die Reporte anhand der Formulardefinition automatisch generiert. Es ist aber möglich, die Reporte per Konfiguration zu verändern.

Speicherort	XML-Datei im Ordner **xml**
Dokumente	Kapitel **Reporte hinzufügen / anpassen** (Seite 68)
Beispiel	ModifyObject_AddSimpleAttribute \WEB-INF\config\custom\xml\custom.xml: Modify Report

4.1.2.3 Attribut zu einer Liste hinzufügen

4.1.2.3.1 Liste anpassen

Die Modifikation der Liste ist der Modifikation des Formulars sehr ähnlich.

Speicherort	XML-Datei im Ordner **xml**.
Vorgehen	1. Fügen Sie ein neues Element **<column>** an der gewünschten Stelle hinzu. 2. Geben Sie der Eigenschaft **id** einen für die Liste eindeutigen Namen. Die Eigenschaft **attribute.idref** muss auf eine gültige Spalte der Datenermittlung verweisen. 3. Fügen Sie ein neues Element **<listHeader>** hinzu. Um eine Sortierung zu ermöglichen, muss die Eigenschaft **column** auf den zugehörigen Spaltennamen zeigen. 4. Füllen Sie den **Property-Key** und die Eigenschaft **width** aus. Der Wert für **width** wird meist in relativen Breiten angegeben und sollte sich auf maximal 98 % summieren. Die übrigen 2 % werden von den Schaltflächen am Beginn und am Ende jeder Zeile eingenommen.
Dokumente	lists_[module].xml, lists.xsd
Beispiel	ModifyObject_AddSimpleAttribute \WEB-INF\config\custom\xml\custom.xml: Modify List

4.1.2.3.2 Properties für Liste hinzufügen / anpassen

Siehe **Properties hinzufügen / anpassen (Seite 23).**

Beispiel

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

4.1.2.3.3 Datenermittlung für Liste anpassen

In der Liste ist eine View als Datenquelle definiert. Diese View muss nun angepasst werden, damit das neue Attribut enthalten ist.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	1. Fügen Sie ein neues Element **<viewColumn>** als Unterelement in das Element **<viewObject>** ein. Das View-Object definiert den Objekttyp der View-Columns. 2. Ordnen Sie der Eigenschaft **id** einen eindeutiger Namen zu. Die Eigenschaft **attributeType** referenziert ein Attribut des zugehörigen Objekttyps. Soll die Spalte nicht nur gefiltert sondern auch ausgegeben werden, muss die Eigenschaft **isSelectColumn** auf **true** gesetzt werden.
Dokumente	view_[module].xml, views.xsd
Beispiel	ModifaObject_AddSimpleAttribute \WEB-INF\config\custom\xml\custom.xml: Modify View

4.1.2.3.4 Renderer hinzufügen

Für alle Attribute ist eine Standdarddarstellung definiert. Um sie zu verändern, müssen Sie eine andere Klasse spezifizieren, welche diese Darstellung generiert oder mittels Standard-Renderer über Parameter die Darstellung anpassen.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Ordnen Sie der Eigenschaft **template** der **<column>** einen gültigen und registrierten Renderer zu.
Dokumente	uiClassMappings.xml, uiClassMappings.xsd
Beispiel	ModifyObject_AddSimpleAttribute \WEB-INF\config\custom\xml\custom.xml: Assign List Renderer

4.1.2.3.5 Reporte hinzufügen / anpassen

Siehe **Reporte hinzufügen / anpassen** (Seite 68).

4.1.2.4 Attribut zu einem Filter hinzufügen

4.1.2.4.1 Listenfilter anpassen

Alle Spalten der Datenermittlung, unabhängig davon ob im Ergebnis vorhanden oder nicht, können gefiltert werden. Die Filterwerte können von den Benutzern dann verändert werden und zwecks häufiger Wiederverwendung auch gespeichert werden.

Es gibt zwei Ansichten für den Filter. Zum einen die Ansicht als einfache Liste im Dialog**Filter Speichern / Konfigurieren**, ähnlich einem Objektformular, und zum anderen die Ansicht als ausklappbaren Filter direkt über der Liste. Das Format für den Dialog **Filter Speichern / Konfigurieren** wird bei der Definition des Filters festgelegt. Die Filterdefinition ist an die Formulardefinition angelehnt und funktioniert nach den gleichen Prinzipien.

Für den ausklappbaren Filter kann die Reihenfolge und Anordnung der einzelnen filterbaren Attribute nochmals angepasst werden. Alle nicht explizit angeordneten Filterattribute werden im erweiterten Teil des Filters in der Reihenfolge der Filterdefinition ans Ende angefügt.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	1. Kopieren Sie den entsprechenden Standardfilter in das Customizing. 2. Fügen Sie ein neues Element **<filterRow>** an der gewünschten Stelle in der Filterdefinition ein. 3. Legen Sie unter dem neuen Element ein Unterelement **<filterElement>** an und setzen Sie die Eigenschaft **dataReference.idref** auf die gewünschte viewColumn. Die Eigenschaften **template** und **filterType** werden automatisch bestimmt, können aber auch direkt gesetzt werden, falls eine abweichende Darstellung (**template**) oder Datenhaltung (**filterType**) gewünscht bzw. erforderlich ist.
Dokumente	filters_[module].xml, uiClassMapping.xml, blClassMapping.xml, filter.xsd, uiClassMapping.xsd, blClassMapping.xsd
Beispiel	ModifyObject_AddSimpleAttribute \WEB-INF\config\custom\xml\custom.xml: Modify filter

4.1.2.4.2 Properties für Filter hinzufügen / anpassen

Siehe **Propertys hinzufügen / anpassen** (Seite 23).

Beispiel

ModifyObject_AddSimpleAttribute

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

4.1.2.4.3 Renderer zuweisen

Siehe **Renderer zuweisen (Formular)** (Seite 26).

4.1.3 Enumeration-Attribut hinzufügen / ändern

4.1.3.1 Enumeration-Attribut anlegen

4.1.3.1.1 Enumeration hinzufügen / anpassen

Für ein neues Enumeration-Attribut kann sowohl eine bestehende Enumeration wiederverwendet oder eine neue angelegt werden. Wir empfehlen eine neue Enumeration anzulegen. Wenn Sie eine bestehende Enumeration wiederverwenden, besteht die Gefahr, dass ein Konflikt mit anderen Enumeration-Attributen entsteht, die auf diese Enumeration verweisen, falls die Elemente angepasst werden müssen.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	1. Fügen Sie ein neues Element **<enum>** ein. 2. Legen Sie die notwendigen Eigenschaften **id** (eindeutiger Name), **isMultiple** (Ein- oder Mehrfachselektion), sowie **type** (**number** oder **string**) fest. 3. Als Unterelemente fügen Sie dann die Elemente der Enumeration als Element **<enumitem>** hinzu. 4. Vergeben Sie für jedes Element **<enumitem>** einen eindeutigen Namen (**id**) und einen entsprechenden Wert (**value**).
Bemerkung	Ist als type **number** gewählt, darf die Eigenschaft **value** der Elemente **<enumitem>** nur Zahlen enthalten. Ist die Eigenschaft **formRelevant** auf **false** gesetzt, kann der Wert nicht im Formular ausgewählt werden. Dies gilt auch für **filterRelevant** hinsichtlich der Wählbarkeit im Filter.
Dokumente	enumerations.xml
Beispiel	ModifyObject_AddEnumerationAttribute \WEB-INF\config\custom\xml\custom.xml: Add/Modify enumeration

4.1.3.1.2 Propertys für Enumeration hinzufügen / anpassen

Siehe **Propertys hinzufügen / anpassen** (Seite 23).

Bemerkung	Per Konvention ergibt sich der Property-Key für ein Enumeration-Element: enumeration.[Name der Enumeration].[Name des Elements].DBI
Beispiel	ModifyObject_AddEnumerationAttribute \WEB-INF\config\custom\properties\application\custom.properties: Add/Modify enumeration

4.1.3.1.3 Objekttyp anpassen

Siehe **Objekttyp anpassen** (Seite 23).

Vorgehen	Ergänzend zu den einfachen Attributen, müssen Sie die Eigenschaft **enumeration** pflegen. Diese gibt an, welche Enumeration für das Attribut verwendet werden soll.
Beispiel	ModifyObject_AddEnumerationAttribute \WEB-INF\config\custom\xml\custom.xml: Add enumeration attribute

4.1.3.1.4 Attribut zu einem Formular hinzufügen

Siehe **Attribut zu einem Formular hinzufügen** (Seite 25).

Beispiel

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 Datenermittlung für Liste anpassen

Siehe **Datenermittlung für Liste anpassen** (Seite 28).

Beispiel

ModifyObject_AddEnumerationAttribute

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

4.1.3.1.6 Attribut zu einer Liste hinzufügen

Siehe **Attribut zu einer Liste hinzufügen** (Seite 27).

Beispiel

ModifyObject_AddEnumerationAttribute

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

4.1.3.1.7 Attribut zu einem Filter hinzufügen

Siehe **Attribut zu einem Filter hinzufügen** (Seite 29).

Beispiel

ModifyObject_AddEnumerationAttribute

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

4.1.4 List-Attribut hinzufügen / ändern

4.1.4.1 List-Attribut anlegen

4.1.4.1.1 Objekttyp anpassen

Siehe **Objekttyp anpassen** (Seite 23).

Bemerkung	List-Attribute benötigen zusätzliche Eigenschaften: ▪ **maxSize** gibt an, wie viele Objekte dem Attribut maximal zugeordnet werden können. **-1** bedeutet hier unbegrenzt viele. ▪ Die Eigenschaft **linkType** benötigt einen eindeutigen numerischen Wert. ▪ **objectType.idref** enthält durch Kommas getrennt die Namen aller Objekttypen, die zugeordnet werden dürfen. In der Regel sollte dies nur ein Objekttyp sein. ▪ Die Eigenschaft **orderType** gibt an, wie zugeordnete Objekte sortiert werden sollen.
Dokumente	objectTypes.xsd
Beispiel	ModifyObject_AddListAttribute \WEB-INF\config\custom\xml\custom.xml: Add list attribute

4.1.4.1.2 Properties hinzufügen / anpassen

Siehe **Properties hinzufügen / anpassen** (Seite 23).

Beispiel

ModifyObject_AddListAttribute

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

4.1.4.1.3 Listenbeschränkungen anpassen

Neben der Beschränkung auf den Objekttyp gibt es noch eine Möglichkeit, die erlaubte Menge an Objekten weiter einzuschränken. Und zwar auf Objekte, deren Attribute bestimmte Werte besitzen.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Fügen Sie ein Element **<listRestriction>** als neues Unterelement hinzu. Es hat keine Eigenschaft sondern dient nur der Gruppierung der Unterelemente **<attributeRestriction>**. Diese haben zwei Eigenschaften die beide vorhanden sein müssen. **attribute** verweist auf ein Attribut des im

	List-Attribute erlaubten Objekttyps. **value** verweist auf die erlaubten Werte des Attributs. Objekte bei denen dieser Wert anders ist, können nicht zugeordnet werden.
Bemerkung	Es sollte nur auf Attribute verwiesen werden, die sich im Lebenszyklus des Objekts nicht mehr verändern, wie der Typ eines Hierarchieobjekts. Elemente **<listRestriction>** eines List-Attributs sind ODER-verknüpft. Elemente **<attributeRestriction>** einer List-Restriction sind Und-verknüpft.
Dokumente	objectTypes.xsd
Beispiel	ModifyObject_AddListAttribute \WEB-INF\config\custom\xml\custom.xml: Add list restrictions

4.1.4.1.4 Rollen anpassen

Für List-Attribute gibt es eine gesonderte Rechtevergabe. Hier wird festgelegt, welche Rolle welche List-Attribute bearbeiten darf. Die Rechte für das Hinzufügen bzw. das Entfernen von Objekten zum List-Attribute können einzeln gesetzt werden. Standardmäßig hat keine Rolle Rechte hierzu, alle Rollen müssen diese Rechte explizit zugewiesen bekommen.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	1. Fügen Sie ein Element **<roles>** hinzu. 2. Geben Sie ihm ein oder mehrere Unterelemente **<role>**, welche aus der Standardkonfiguration übernommen werden. 3. Passen Sie dann die Rechte an. In diesem Fall brauchen die Rollen Rechte zum Hinzufügen und Entfernen von Objekten zum neuen ListAttribut. 4. Geben Sie dem Element **<object>**, welches den passenden Objekttyp referenziert, ein neues Unterelement **<relation>**. Die Eigenschaft **right.idref** kennt die möglichen Werte **a** (attach = hinzufügen), **r** (remove = entfernen) und **ar** (attach/remove = hinzufügen/entfernen). listAttrType.id referenziert das entsprechende List-Attribut.
Dokumente	roles.xml, roles.xsd
Beispiel	ModifyObject_AddListAttribute \WEB-INF\config\custom\xml\custom.xml: Modify roles

4.1.4.1.5 Attribut zu einem Formular hinzufügen

Siehe **Attribut zu einem Formular hinzufügen** (Seite 25).

Vorgehen	Für List-Attribute gibt es zusätzliche Eigenschaften, die gepflegt werden müssen. Soll das List-Attribut durch den Benutzer editierbar sein, wird eine Auswahlliste benötigt, welche die zuweisbaren Objekte zur Auswahl anbietet. Hier gibt es zwei Möglichkeiten. Sie können sich eine Auswahlliste automatisch generieren lassen oder selbst eine Liste definieren (siehe **Auswahlliste hinzufügen** (Seite 36)). Die zweite Möglichkeit ist vor allem dann relevant, wenn die Auswahlliste Sonderfälle enthält, welche die automatischen Listen nicht abdecken können. Konfiguriert wird die Auswahlliste über ein Unterelement **<parameter>** mit dem Namen **selectionList** und dem Namen der Auswahlliste als **value**. Für eine automatisch generierte Auswahlliste muss **value** den Wert **AUTO** besitzen. Des Weiteren können die Schaltflächen zur Verwaltung und Manipulation der Objekte angepasst werden. **Öffnen**, **Bearbeiten**, **Zuordnung anlegen** und **Zuordnung** entfernen sind die Standardschaltflächen. Hierzu gibt es die Unterelemente **<button.add>** und **<button.remove>**. Die Eigenschaft **button.idref** gibt die Schaltfläche an, welche hinzugefügt oder entfernt werden soll. location gibt an, ob die Schaltfläche in die Symbolleiste (**toolbar**) eingefügt soll oder einzeln für jedes zugeordnete Objekt (**row**) angezeigt werden soll. **type** gibt an, ob die Schaltfläche nur verfügbar ist, wenn das List-Attribut durch den Benutzer bearbeitbar ist (**writable**) oder immer (**always**).
Bemerkung	Eine automatische Auswahlliste berücksichtigt den Mandanten des Objektes, die List-Restriction des List-Attributs sowie die bereits zugewiesenen Objekte. Die primäre Spalte der Auswahlliste enthält das Attribut welches in der Eigenschaft **nameAttribute** des Auswahlobjekttyps angegeben ist. Die übrigen Spalten und Filter ergeben sich aus den Attributen, welche in der Eigenschaft **descriptionAttributes** angegeben sind.
Dokumente	objectTypes.xsd
Beispiel	ModifyObject_AddListAttribute \WEB-INF\config\custom\xml\custom.xml: Modify form

4.1.4.2 Auswahlliste hinzufügen

4.1.4.2.1 Auswahlliste anpassen

Das Hinzufügen einer Auswahlliste ähnelt dem Anlegen einer normalen Liste. Dies beinhaltet eine Datenermittlung, eine Darstellung der Liste und eine Filterdefinition. Um aus dieser normalen Liste eine Auswahlliste zu machen, müssen nur noch einige Eigenschaften gesetzt werden.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	1. Legen Sie die Liste an und pflegen Sie folgende Eigenschaften des Elements **<list>**: a. **listType** erhält den Wert **selection**. b. destDataType**.idref** erhält als Wert den Objekttyp des List-Attributs. c. destAttributeType**.idref** erhält den Namen des List-Attributs selbst.
Dokumente	lists.xsd
Beispiel	ModifyObject_AddListAttribute \WEB-INF\config\custom\xml\custom.xml: Add selection list

4.1.4.2.2 Properties hinzufügen / anpassen

Siehe **Properties hinzufügen / anpassen** (Seite 23).

Bemerkung	Das Weglassen der rechten Seite in der Property-Datei sorgt dafür, dass keine Hilfe-Schaltfläche für dieses Element angezeigt wird.
Beispiel	ModifyObject_AddListAttribute \WEB-INF\config\custom\properties\help\custom_helpids.properties: Add selection list

4.1.4.2.3 Datenabfrage für Auswahlliste anpassen

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	1. Ergänzen Sie die Datenabfrage, neben den sonstigen Bedingungen, um zwei Elemente **<viewCondition>**. 2. Setzen Sie eine Bedingung auf das Attribut **obj_id** des Hauptobjekttyps mit dem Compare-Type **NOTIN**. **currentObjectType.id** und **currentAttributeType.id** verweisen wieder auf das List-Attribut und dessen Objekttypen. Die zweite Bedingung filtert das Attribut **client_sign**. **currentObjectType.id** erhält den gleichen Wert wie zuvor, **currentAttributeType**. **id** erhält den Wert **client_sign**.
Dokumente	views.xsd
Beispiel	ModifyObject_AddListAttribute \WEB-INF\config\custom\xml\custom.xml: Add selection view

4.1.4.2.4 Renderer zuweisen

Siehe **Renderer zuweisen** (Seite 26).

4.1.4.2.5 Auswahllistenfilter hinzufügen

Siehe **Attribut zu einem Filter hinzufügen** (Seite 29).

Bemerkung	Hier gibt es keinen Unterschied zu einem normalen Filter.
Dokumente	filters.xsd
Beispiel	ModifyObject_AddListAttribute \WEB-INF\config\custom\xml\custom.xml: Add selection filter

4.2 Objekt-Lebenszyklus anpassen

Der Objekt-Lebenszyklus wird im Wesentlichen durch die folgenden Konfigurationsdateien gesteuert bzw. konfiguriert:

- Workflow-Konfiguration: XML-Dateien (workflow_*.xml) im Order **xml.**
- Command-Chain-Katalog-Konfiguration: XML-Dateien (commandChains_*.xml) im Ordner **xml.**
- Regel-Konfiguration: DRL-Dateien im Ordner **rules** (siehe Kapitel **Regel hinzufügen / anpassen** (Seite 77)).
- Generator: Zeitgesteuertes Generieren von Objekten durch das Ausführen der Transitions **<prepare>** und **<insert>**.
- Monitor: Zeitgesteuerte Prüfung der Bearbeitungszeiträume und Änderung der Attributwerte (siehe Kapitel **Monitor** (Seite 83)).

Der Zusammenhang von Workflow-Konfiguration, den Regeln und den Command-Chains ist wie folgt:

- Die Regeln definieren die Attributzustände, z. B. änderbar, sichtbar, die für ein Objekt in einem bestimmten Zustand gelten. Dazu kann die State-ID in den Regeln verwendet werden. Details siehe **Regel hinzufügen / anpassen** (Seite 77)).
- Die Command-Chains enthalten die Commands, die bei der Ausführung einer Transition ausgeführt werden.

In den folgenden Kapiteln wird die Workflow-, Command-Chain- sowie Regel-Konfiguration erläutert.

4.2.1 Workflow-Konfiguration

Ein Workflow besteht aus Status und Transitions. Ein Status repräsentiert einen Zustand eines Objekts. Ein solcher Zustand wird über Attributwerte eines Objekts definiert. Jeder Status muss über mindestens eine Transition erreichbar sein.

Eine **Transition** repräsentiert einen Übergang von einem Zustand in einen anderen. Zu jedem Zustandsübergang muss eine Command-Chain definiert sein. Diese Command-Chain wird beim Zustandsübergang ausgeführt. Eine Command-Chain enthält eine beliebige Anzahl von Commands. Leere Chains sind auch erlaubt.

Beim Anpassen eines Workflows genügt es, ausschließlich die neuen oder geänderten Status aufzuführen. Alle unveränderten Status können weggelassen werden.

Im Folgenden wird das Hinzufügen von Status und Transitions beschrieben. Details zur Konfiguration einer Command-Chain finden Sie unter **Konfiguration des Command Chain-Katalogs** (Seite 49).

4.2.1.1 Status hinzufügen

Ein Workflow kann aus bis zu vier verschiedenen Statustypen bestehen. Die beiden folgenden dürfen jeweils nur einmal pro Workflow vorhanden sein:

- Ein Status, der ein neues, unbearbeitetes, nicht gespeichertes Objekt repräsentiert (**<state.initial>**).
- Ein Status, der ein noch nicht gespeichertes Objekt repräsentiert, das bereits von einem Benutzer oder vom System bearbeitet wurde (**<state.prepared>**).

Diese können mehrfach in einem Workflow enthalten sein, müssen aber eindeutige IDs haben:

- Ein Status, der ein aktives Objekt repräsentiert (**<state>**) oder
- ein Status, der ein gelöschtes Objekt repräsentiert (**<state.deleted>**).

4.2.1.1.1 Status für aktives Objekt hinzufügen

Ein Objekt ist aktiv, wenn es persistent und nicht gelöscht ist, also z. B. durch einen Benutzer in einem Formular bearbeitet werden kann.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Kopieren Sie ein Element <workflow> aus dem Standard in das Customizing. Danach können Sie ein oder mehrere neue <states> innerhalb des Elements **<workflow>** anlegen. Das Attribut **id** muss einen Wert haben, der innerhalb des Workflows eindeutig ist.
Bemerkung	Ein Status eines Objekts muss durch die Werte der Elemente **<Attribute>** eindeutig identifizierbar sein. D. h., ein Objekt ist in einem bestimmten Status, wenn es die in den Elementen **<Attribute>** angegebenen Werte hat. Zur Ermittlung des Zustands werden die persistenten Werte eines Objekts verwendet. Ein hinzugefügter Status muss durch mindestens eine Transition erreichbar sein. Siehe dazu **Transition hinzufügen** (Seite 40).
Dokumente	workflow_*.xml, workflow.xsd
Beispiel	ModifyObjectLifecycle \WEB-INF\config\custom\xml\testcase_workflow_custom.xml: New custom active workflow state

4.2.1.1.2 Status für ein gelöschtes Objekt hinzufügen

Ein Objekt kann von einem Benutzer durch Klick auf **Löschen** deaktiviert/gelöscht werden. Handelt es sich bei dem Objekt um ein versioniertes Objekt, wird es nicht aus der Datenbank gelöscht sondern nur deaktiviert. Ein deaktiviertes Objekt kann wieder reaktiviert werden. Siehe dazu **Transition hinzufügen (Seite 40)** und **Recover-Transition hinzufügen** (Seite 46).

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Kopieren Sie ein Element **<workflow>** aus dem Standard in das Customizing. Danach können Sie ein oder mehrere neue **<state.deleted>** innerhalb des Elements **<workflow>** unterhalb der aktiven Status anlegen. Das Attribut **id** muss einen Wert haben, der innerhalb des Workflows für gelöschte Status eindeutig ist.
Bemerkung	Ein Status eines Objekts muss durch die Werte der Elemente **<Attribute>**eindeutig identifizierbar sein. D. h., ein Objekt ist in einem bestimmten Status, wenn es die in den Elementen **<Attribute>** angegebenen Werte hat. Zur Ermittlung des Status werden die persistenten Werte eines Objekts verwendet. Ein hinzugefügter state.deleted muss durch mindestens eine Delete-Transition erreichbar sein. Siehe dazu **Delete-Transition hinzufügen** (Seite 45).
Dokumente	workflow_*.xml, workflow.xsd
Beispiel	ModifyObjectLifecycle \WEB-INF\config\custom\xml\testcase_workflow_custom.xml: New custom deleted workflow state

4.2.1.2 Transition hinzufügen

Abhängig vom Status eines Objektes gibt es unterschiedliche Transitions die ausgeführt werden können. Mögliche Transitions beim aktiven Zustand (**<state>**) sind: **update**, **reset** und **delete**. Im deaktivierten Zustand gibt es nur die Transition **recover**, um den Zustand zu verlassen bzw. in einen aktiven Zustand zurückzukehren. Um bereits bei der Erzeugung eines Objektes unterschiedliche Transitions ausführen zu können, gibt es die beiden Zustände **initial.state** und **prepare.state** mit den beiden Transitions **prepare** und **insert**.

4.2.1.2.1 Prepare-Transition hinzufügen

Eine prepare-Transition wird ausgeführt, wenn ein Objekt von einem Benutzer oder von einem Job erzeugt wird. Nach Ausführung dieser Transition ist das Objekt immer im Zustand state.prepared.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	1. Kopieren Sie ein Element **<workflow>** aus dem Standard in das Customizing. 2. Legen Sie eine neue Prepare-Transition innerhalb des Elements **<state.initial>** an. Ein Prepare-Element muss im Attribut **chain.id** auf eine existierende Command-Chain des zum Workflow gehörenden Command-Kataloges verweisen (**commandChains_*.xml**).
Bemerkung	Verschiedene Prepare-Transitions müssen sich durch eines die beiden möglichen untergeordneten Elemente **<permission.workflow>** und **<permission.job>** unterscheiden. ▪ **<permission.job>** wird verwendet, um ein Transition nur vom angegebenen Job ausführen zu können. ▪ **<permission.workflow>** wird verwendet, um eine Transition nur vom angegebenen Workflow ausführen zu können.
Dokumente	workflow.xsd, workflow_*.xml, commandChains_*.xml
Beispiel	ModifyObjectLifecycle \WEB-INF\config\custom\xml\user_workflow_custom.xml: New custom prepare transition

4.2.1.2.2 Insert-Transition hinzufügen

Eine Insert-Transition mit dem Status **state.prepared** wird beim ersten Speichern des Objektes ausgeführt. Nach Ausführung der Transition muss sich das Objekt in einem im Workflow definierten Zustand befinden.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	1. Kopieren Sie ein Element **<workflow>** aus dem Standard in das Customizing. 2. Legen Sie eine neue Insert-Transition innerhalb des Elements **<state.prepared>** an. Ein Insert-Element muss jeweils im Attribut **to.state.id** auf einen im Workflow existierenden aktiven Status verweisen, sowie im Attribut **chain.id** auf eine existierende Command-Chain des zum Workflow gehörenden Command-Kataloges (**commandChains_*.xml**).
Bemerkung	Verschiedene Insert-Transitions müssen sich durch eines die beiden möglichen untergeordneten Elemente **<permission.workflow>** und **<permission.job>** unterscheiden. ▪ <permission.job> wird verwendet um eine Transition nur vom angegebenen Job ausführen zu können. ▪ <permission.workflow> wird verwendet um eine Transition nur vom angegebenen Workflow ausführen zu können.
Dokumente	workflow.xsd, workflow_*.xml, commandChains_*.xml
Beispiel	ModifyObjectLifecycle \WEB-INF\config\custom\xml\user_workflow_custom.xml: New custom insert transition

4.2.1.2.3 Update-Transition hinzufügen

Eine Update-Transition wird ausgeführt, wenn ein Objekt vom Benutzer in einem Formular oder von einem internen Prozess, z. B. Job, gespeichert wird. Ein Objekt kann nur dann gespeichert werden, wenn es in einem Zustand ist, der eine Update-Transition enthält. Die Funktion Speichern ist bei Objekten in einem Zustand ohne ausgehende Update-Transition nicht aktiv.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	1. Kopieren Sie ein Element **<workflow>** aus dem Standard in das Customizing. 2. Legen Sie eine neue oder mehrere Update-Transitions innerhalb des Elements **<transitions>** an. Ein Update-Element muss jeweils im Attribut **to.state.id** auf einen im Workflow existierenden aktiven Status verweisen, sowie im Attribut **chain.id** auf eine existierende Command-Chain des zum Workflow gehörenden Command-Kataloges (commandChains_*.xml).
Bemerkung	Eine hinzugefügte Update-Transition muss auf einen existierenden Status (**to.state.id**) und eine Command-Chain (**chain.id**) verweisen.
Dokumente	workflow.xsd, workflow_*.xml, commandChains_*.xml
Beispiel	ModifyObjectLifecycle \WEB-INF\config\custom\xml\testcase_workflow_custom.xml: New custom update transition

4.2.1.2.4 Reset-Transition hinzufügen

Eine reset-Transition wird ausgeführt, wenn ein Benutzer in einem Formular auf **Zurücksetzen** klickt oder wenn ein interner Prozess das Objekt zurücksetzt. Ein Objekt kann nur dann zurücksetzt werden, wenn es in einem Zustand ist der eine Reset-Transition enthält. Die Funktion Zurücksetzen ist bei Objekten in einem Zustand ohne ausgehende Update-Transition nicht aktiv.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	1. Kopieren Sie ein Element **<workflow>** aus dem Standard in das Customizing. 2. Legen Sie eine oder mehrere neue Reset-Transitions innerhalb des Elements **<transitions>** Elements an. Ein Reset-Element muss jeweils im Attribut **to.state.id** auf einen im Workflow existierenden aktiven Status verweisen, sowie im Attribut **chain.id** auf eine existierende Command-Chain des zum Workflow gehörenden Command-Kataloges (commandChains_*.xml).
Bemerkung	Eine hinzugefügte Reset-Transition muss auf einen existierenden Status (**to.state.id**) und eine Command-Chain (**chain.id**) verweisen.
Dokumente	workflow.xsd, workflow_*.xml, commandChains_*.xml
Beispiel	ModifyObjectLifecycle \WEB-INF\config\custom\xml\testcase_workflow_custom.xml: New custom reset transition

4.2.1.2.5 Delete-Transition hinzufügen

Eine Delete-Transition wird ausgeführt, wenn ein Benutzer in einem Formular auf **Löschen** klickt oder wenn ein interner Prozess ein Objekt löscht. Ein Objekt kann nur dann gelöscht werden, wenn es in einem Zustand ist, der eine Delete-Transition enthält. Die Funktion Löschen ist bei Objekten in einem Zustand ohne ausgehende Delete-Transition nicht verfügbar.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	1. Kopieren Sie ein Element **<workflow>** aus dem Standard in das Customizing. 2. Legen Sie eine oder mehrere neue Delete-Transitions innerhalb des Elements **<transitions>** an. Ein Element **delete** muss jeweils im Attribut **to.state.id** auf einen im Workflow existierenden aktiven Status verweisen, sowie im Attribut **chain.id** auf eine existierende Command-Chain des zum Workflow gehörenden Command-Kataloges (**commandChains_*.xml**).
Bemerkung	Eine hinzugefügte Delete-Transition muss auf einen existierenden Status (**to.state.id**) und eine Command-Chain (**chain.id**) verweisen.
Dokumente	workflow.xsd, workflow_*.xml, commandChains_*.xml
Beispiel	ModifyObjectLifecycle \WEB-INF\config\custom\xml\testcase_workflow_custom.xml: New custom delete transition

4.2.1.2.6 Recover-Transition hinzufügen

Eine recover-Transition wird ausgeführt, wenn ein Benutzer in einem Formular eines gelöschten bzw. deaktivierten Objekts auf **Wiederherstellen** klickt oder wenn ein interner Prozess das Objekt wiederherstellt. Ein Objekt kann nur dann wiederhergestellt werden, wenn es in einem inaktiven Zustand ist, der eine Recover-Transition enthält. Andere Zustände können keine ausgehende Recover-Transition haben. Die Funktion Wiederherstellen ist bei inaktiven Objekten ohne ausgehende Recover-Transition nicht verfügbar.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	1. Kopieren Sie ein Element **<workflow>** aus dem Standard in das Customizing. 2. Legen Sie eine oder mehrere Delete-Transitions innerhalb des Elements **<transitions>** an. Ein Recover-Element muss jeweils im Attribut **to.state.id** auf einen im Workflow existierenden inaktiven Status, sowie im Attribut **chain.id** auf eine existierende Command-Chain des zum Workflow gehörenden Command-Katalogs (**commandChains_*.xml**) verweisen.
Bemerkung	Eine hinzugefügte recover-Transition muss auf einen existierenden aktiven Status (**to.state.id**) und eine Command-Chain (**chain.id**) verweisen.
Dokumente	workflow.xsd, workflow_*.xml, commandChains_*.xml
Beispiel	ModifyObjectLifecycle \WEB-INF\config\custom\xml\testcase_workflow_custom.xml: New custom recover transition

4.2.1.3 Task Items modifizieren

In ARIS Risk & Compliance Manager werden transaktionale Objekte während ihres Workflows durch Tasks begleitet. Ein Task repräsentiert in diesem Zusammenhang die Information, welche Benutzer oder Benutzergruppen momentan für die Bearbeitung des betroffenen transaktionalen Objekts zuständig sind. Es gibt außerdem den Bearbeitungs- und Kontrollzeitraum dieses Tasks aus. Innerhalb der Standardkonfiguration sind diese Zeiträume immer mit denen des transaktionalen Objektes identisch.

Task-Items sind immer mit genau einem Workflow-Zustand verknüpft. Verlässt das transaktionale Objekt diesen Workflow-Zustand, enden auch die entsprechenden Tasks. Gleichzeitig werden alle neuen Tasks gestartet, die für den neuen Workflow-Zustand definiert sind.

Task-Items werden in ARIS Risk & Compliance Manager für zwei Aspekte verwendet:

- Sie werden für jeden Benutzer als Liste direkt nach dem Anmelden angezeigt (**Home > Meine Aufgaben**).
- Sie werden zur Konfiguration des Überwachungs-Jobs verwendet. Dieser prüft für alle Task-Items, welcher Zeitanteil des darin definierten Arbeitszeitraums bereits verstrichen ist und reagiert dann gemäß den definierten Eskalationen (Seite 83).

Einem Workflow-Zustand können beliebig viele Task-Items zugewiesen werden. Wenn sich ein Objekt innerhalb seines Workflows in einem Zustand befindet, an dem kein Task-Item definiert ist, wird es vom Überwachungs-Job nicht beachtet.

Für Task-Items können die folgenden Attribute gesetzt werden.

Role

Role bestimmt, welche Eskalationen der Überwachungs-Job bei der Bearbeitung des Task-Items heranzieht. Falls der Wert gesetzt ist, muss er einer Rollen-ID aus der Datei **roles.xml** entsprechen. Falls er nicht gesetzt ist, werden die definierten Eskalationen ohne Rolleneinschränkung verwendet.

Neben Role müssen auch der definierte Objekttyp der Eskalation sowie der Name übereinstimmen, falls am Task-Item angegeben.

Hinweis

Die Verwendung dieses Attributs ist wegen der Kompatibilität noch möglich, sollte aber bei der Anpassungen von Task-Items vermieden werden. Es wird in künftigen Versionen nicht mehr unterstützt. Stattdessen wird die Nutzung des unten aufgeführten Attributs **Name** empfohlen.

Name

Name bestimmt, welche Eskalationen der Überwachungs-Job bei der Bearbeitung des Task-Items heranzieht. Falls er nicht gesetzt ist, werden die definierten Eskalationen ohne Namensangabe verwendet.

Neben Name müssen auch der definierte Objekttyp der Eskalation sowie Role übereinstimmen, falls am Task-Item angegeben.

State

Es gibt drei Zustände, die ein Task-Item haben kann. Folgendes geschieht, wenn ein transaktionales Objekt einen Workflow-Zustand mit einem dort definierten Task-Item erreicht:

- **open**: Solange sich das transaktionale Objekt im Workflow-Zustand mit diesem Task-Item befindet, prüft der Überwachungs-Job bei jedem Lauf, welche Eskalationsstufe aktuell erreicht wurde.
- **completed**: Solange sich das transaktionale Objekt im Workflow-Zustand mit diesem Task-Item befindet, wird der Überwachungs-Job es ignorieren.
- **not_completed**: Solange sich das transaktionale Objekt im Workflow-Zustand mit diesem Task-Item befindet, wird der Überwachungs-Job es ignorieren.

Planned start date

Gibt das Attribut des transaktionalen Objekts an, dessen Wert als Startdatum des Bearbeitungszeitraums verwendet wird. Falls es nicht gesetzt ist, wird **plannedstartdate** verwendet.

Planned end date

Gibt das Attribut des transaktionalen Objekts an, dessen Wert als Enddatum des Bearbeitungszeitraums verwendet wird. Falls es nicht gesetzt ist, wird **plannedenddate** verwendet.

Control start date

Gibt das Attribut des transaktionalen Objekts an, dessen Wert als Startdatum des Kontrollzeitraums verwendet wird. Falls es nicht gesetzt ist, wird **controlstartdate** verwendet.

Control end date

Gibt das Attribut des transaktionalen Objekts an, dessen Wert als Enddatum des Kontrollzeitraums verwendet wird. Falls es nicht gesetzt ist, wird **controlenddate** verwendet.

Responsible

Gibt die Verantwortlichen für den Task an. Wenn dieses Attribut gesetzt ist, muss sein Wert der Attribut-ID eines Listenattributs entsprechen, dessen Elemente vom Typ **USERGROUP** oder **USER** sind.

Time limitation

Legt fest, ob der Task eine zeitliche Begrenzung und damit eine Relevanz für den Überwachungs-Job haben soll oder nicht. Hat dieses Attribut den Standardwert **true**, werden der Bearbeitungszeitraum und die Kontrollzeitraum gemäß der oben beschriebenen Attribute gesetzt. Ist **false** gesetzt, wird der Beginn des Arbeitszeitraums auf **Now** gesetzt. Das Ende des Arbeitszeitraums und des Kontrollzeitraums werden nicht gesetzt.

4.2.2 Command-Chain-Katalog konfigurieren

Die Command-Chains (im Folgenden als Chains bezeichnet), welche durch einen bestimmten Workflow ausgeführt werden können, sind jeweils in einem Command-Chain-Katalog (im Folgenden als Katalog bezeichnet) zusammengefasst. Der zum Workflow gehörende Katalog hat die gleiche ID wie der entsprechende Workflow. Ein Katalog kann um neue Chains ergänzt werden. Eine Chain muss nicht von einer Transition verwendet werden. Das Löschen von nicht verwendeten Chains ist nicht notwendig. Der Inhalt einer Chain kann geändert werden, d. h., Commands können hinzugefügt oder entfernt werden. Beim Anpassen eines Katalogs genügt es, ausschließlich die neuen oder geänderten Chains aufzuführen. Alle unveränderten Chains können weggelassen werden.

4.2.2.1 Command-Chain ändern

Wenn beim Ausführen einer Transition zusätzliche Aktionen ausgeführt werden sollen, z. B. Versenden von Nachrichten, Ändern von Daten usw., kann dies durch Hinzufügen von Commands in eine existierende Chain erreicht werden.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	1. Kopieren Sie ein Element **<catalog>** aus dem Standard in das Customizing. 2. Passen Sie eine oder mehrere Command-Chains innerhalb des Elements **<catalog>** an, indem Sie die Chain um ein Command-Element erweitern oder ein Command-Element entfernen. Jedes Command-Element hat eine ID, welche eine Command-Implementierung repräsentiert. Mögliche IDs finden Sie in der Datei **commandClassMapping.xml**. Details zur Verwendung sowie Parametrisierung der jeweiligen Commands finden Sie im Java-Doc des entsprechenden Commands.
Bemerkung	Durch das Ändern einer Chain ist es möglich, dass der Zielzustand einer Transition aus Sicht der Workflow-Konfiguration nicht mehr gültig ist. **Beispiel** Wenn eine Transition in einem Zustand endet, der durch den Wert **X** des Attributs **A** definiert ist aber ein Command den Wert des Attributs **A** auf **Y** setzt, ist der Zielzustand ungültig bzw. nicht erreicht. In diesem Fall wird die Transition-Ausführung rückgängig gemacht und eine Fehlermeldung angezeigt.
Dokumente	commandChains.xsd, commandChains_*.xml
Beispiel	ModifyObjectLifecycle \WEB-INF\config\custom\xml\testcase_catalog_custom.xml: Change command chain

4.2.2.2 Command-Chain hinzufügen

Wenn ein Workflow um eine neue Transition erweitert wird und keine Chain existiert, die verwendet werden kann, muss eine neue Chain in den Katalog eingefügt werden.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	1. Kopieren Sie ein Element **<catalog>** aus dem Standard in das Customizing. 2. Passen Sie eine oder mehrere Command-Chains innerhalb des Elements **<catalog>** an. 3. Ergänzen Sie die neue Chain mit einer beliebigen Anzahl von Command-Elementen. Eine Chain kann auch leer sein, d. h. kein Command-Element enthalten. Jedes Command-Element hat eine ID, welche eine Command-Implementierung repräsentiert. Mögliche IDs finden Sie in der Datei **commandClassMapping.xml**. Details zur Verwendung sowie Parametrisierung der jeweiligen Commands finden Sie im Java-Doc des entsprechenden Commands.
Bemerkung	Durch das Ändern einer Chain ist es möglich, dass der Zielzustand einer Transition aus Sicht der Workflow-Konfiguration nicht mehr gültig ist. **Beispiel** Wenn eine Transition in einem Zustand endet, der durch den Wert **X** des Attributs **A** definiert ist aber ein Command den Wert des Attributs **A** auf **Y** setzt, ist der Zielzustand ungültig bzw. nicht erreicht. In diesem Fall wird die Transition-Ausführung rückgängig gemacht und eine Fehlermeldung angezeigt.
Dokumente	commandChains.xsd, commandChains_*.xml
Beispiel	ModifyObjectLifecycle \WEB-INF\config\custom\xml\testcase_catalog_custom.xml: Add custom command chain

4.2.3 Benutzerinteraktionen anpassen / hinzufügen

Um eine Command-Chain abarbeiten zu können ist es manchmal erforderlich, zusätzlich Informationen durch den Benutzer einzuholen. Hierzu ist es möglich, eigene Dialoge zu konfigurieren und durch ein Command der Chain aufrufen zu lassen. Hierzu wird die Ausführung der Chain an dieser Stelle unterbrochen, und eine Eingabemaske erscheint. Ist diese Maske korrekt gefüllt worden, beginnt die Command-Chain von Neuem. Diesmal aber ergänzt um die Eingabe des Benutzers. Grundsätzlich gibt es derzeit zwei Arten von Dialogen. Einfache Dialoge verlangen eine einfache Bestätigung oder Entscheidung, komplexe Dialoge haben ein oder mehrere Eingabefelder.

4.2.3.1 Bestätigungsdialoge

Es gibt zwei unterschiedliche Bestätigungsdialoge. Der einfachste Dialog hat die ID **okCancel**, welcher neben einer Frage oder einem Hinweis die beiden Schaltflächen **Ok** und **Abbrechen** bietet. **Abbrechen** beendet die Ausführung der kompletten Command-Chain. Mit **Ok** wird die Chain weiter ausgeführt. Eine Schaltfläche mehr bietet der Dialog mit der ID **yesNoCancel**. Neben dem schon bekannten **Abbrechen** bietet er noch die Möglichkeiten **Ja** und **Nein**, welche beide das Fortsetzen der CommandChain erlauben, aber es den nachfolgenden Commands der Chain ermöglichen, zwei unterschiedliche Wege zu gehen, z. B. kann in einem Fall eine Nachricht gesendet werden, im anderen nicht.

Generell ist es möglich in einer Command-Chain mehrere Dialoge zu verwenden, allerdings müssen alle Dialoge einer Chain eindeutige IDs haben. Informationen dazu, wie man neue Dialoge definiert, u. a. auch Bestätigungsdialoge, für den Fall dass man mehrere in einer Chain braucht, finden Sie nachfolgend.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	1. Erweitern Sie die relevante Command-Chain an der gewünschten Stelle um ein weiteres <command>-Element. 2. Verwenden Sie die ID **requestDialog**. 3. Fügen Sie zwei untergeordnete <parameter>-Elemente hinzu. 4. Das erste Element hat das Name-Attribut mit dem Wert **dialogID** und das value-Attribut einen der beiden Werte **okcancel** oder **yesnocancel**. 5. Das zweite Element hat das Name-Attribut mit dem Wert **propertyKey** und das Value-Attribut erhält als Wert den Property-Key, der die Frage repräsentiert die dem Benutzer gestellt wird.
Bemerkung	Haben Sie einen eigenen Dialog definiert, verwenden Sie dessen ID.
Dokumente	commandChains.xsd, commandChains_*.xml
Beispiel	ModifyObjectLifecycle \WEB-INF\config\custom\xml\testcase_catalog_custom.xml: Add custom dialog

4.2.3.2 Eingabedialoge

Wird mehr als eine einfache Bestätigung benötigt, werden Eingabedialoge verwendet. Damit ist es möglich dem Benutzer ein oder mehrere Eingabefelder zum Ausfüllen bereitzustellen.

Für jedes Eingabefeld wird ein Attribut definiert. Um die unterschiedlichen Anforderungen der Eingabefelder abzubilden, gibt es mehrere Attributtypen, ähnlich wie bei den Objektdefinitionen (**objectTypes.xml**). Auch diese Attribute können mit Validierern (**<validator>**) versehen werden, welche beispielsweise den Gültigkeitsbereich des Attributes einschränken, indem die Länge eines Eingabefeldes beschränkt wird oder bei einem Zahleingabefeld die Menge der gültigen Zahlen. Die Namen der XML–Elemente ergeben sich aus dem Namen des Attributtyps, z. B. **<booleanAttribute>**, **<longAttribute>**. Bei den IDs der Attribute sollten Namenskonventionen verwendet werden, die es erleichtern mit den gesammelten Daten des Dialogs zu arbeiten. Soll beispielsweise ein Attribut des Dialoges in ein Attribut eines Objektes übertragen werden, sollte unbedingt Namensgleichheit bestehen. So heißt das Attribut für die Mandantenauswahl, z. B bei der Objekterstellung, im Dialog **client_sign**, wie das entsprechende Attribut des Objektes das erstellt werden soll.

Bestehende Attributtypen

boolean	Attribut für boolesche Werte (**Ja / Nein**).
date	Attribut für Datumswerte.
double	Attribut für Gleitkommazahlen.
enum	Attribut für Enumerationen (**enumerations_*.xml**).
long	Attribut für Ganzzahlen.
selection	Attribut, um Werte aus einer vorgegebenen dynamischen Liste zu wählen, z. B. für die Wahl des Mandanten. Um die möglichen Werte der Auswahlliste zu bestimmen, gibt es die **PredefinedValueProvider**. Einige dieser konfigurierbaren Provider liefert die Applikation bereits mit (siehe unten). Eigene Implementerungen sind aber auch möglich.
string	Attribut für einfache Texteingabefelder. Inhalt und Länge des Textes können eingeschränkt werden
text	Attribut für mehrzeilige Texte.

Für das Attribut **selection** gibt es mehrere **PredefinedValueProvider**. Sie können parametrisiert werden und bieten verschiedene Möglichkeiten.

Bestehende PredefinedValueProvider

client	Liefert eine Liste aller Mandanten, auf die der Benutzer Zugriff hat. Diese Liste kann durch maximal einen der folgenden Parameter eingeschränkt werden: ▪ **componentRight** Liefert nur Mandanten für die es ein bestimmtes **componentRight** gibt (**roles.xml**). ▪ **objectRight** Liefert nur Mandanten in denen es für den aktuellen **ObjektTyp** (bestimmt über den Parameter **objectTypeId**) ein bestimmtes **objectRight** (**roles.xml**) gibt. ▪ **licensedComponent** Liefert nur Mandanten, welche eine bestimmte lizensierte Komponente besitzen. Zusätzliche Parameter: ▪ **objectTypeId** ID eines Objekttyps (**objecttypes.xml**), die z. B. durch das Command **createObjectDialog** gesetzt wird. Dieser Parameter wird nur im Zusammenhang mit dem Parameter **objectRight** benötigt.
static	Verwaltet eine vorher im Java-code festgelegte Liste von Elementen
usergroup	Liefert eine Liste mit verfügbaren Benutzergruppen. Parameter: ▪ **client_sign** Bestimmt auf welchen Mandanten die Gruppen eingeschränkt werden sollen, wenn **clientDependent=true**. Dieser Parameter kann auch dynamisch über ein anderes Attribut des Dialoges bestimmt werden (siehe Dialog **loss_create**). ▪ **roles** Eine komma-separierte Liste von Rollenbezeichnungen (**roles.xml**). Nur Gruppen mit entsprechenden Rollen werden berücksichtigt. ▪ **clientDependent** Bestimmt ob mandantenspezifische (**true**) oder mandantenübergreifende (**false**) Gruppen gesucht werden (Standard **false**).
view	Verwendet eine View (**views_*.xml**), um eine Liste mit Elementen aufzubauen. Die Parameter **value**, **id** und **client** verlangen als Werte die IDs der entsprechenden Spalten in der View.

Im folgenden Beispiel wird ein neuer Dialog angelegt, welcher beim Erstellen einer neuen Kontrolle erscheint. Dabei werden einige Attribute der Kontrolle abgefragt und anschließend in der neu erstellten Kontrolle gesetzt.

Speicherort	XML-Datei im Ordner xml
Vorgehen	1. Definieren Sie ein neues Element **<dialog>** mit einer eindeutigen ID. 2. Fügen Sie für jedes Attribut des Dialogs ein passendes untergeordnetes Attribut-Element ein. Mögliche Attributtypen finden Sie oben im Text. 3. Vergeben Sie dafür jeweils eine ID. Sollen die Eingabewerte direkt in das Attribut eines Objektes übertragen werden, sollten beide die gleiche ID haben, z. B. **client_sign** (siehe auch **objectTypes.xml**). Jeder Dialog und jedes seiner Attribute erzwingt die Existenz entsprechender Property-Keys **dialog.<dialogID>.DBI** , **attribute.<dialogID>.<attributeID>.DBI**). 4. Legen Sie ein neues Element **<form>** an. Die ID des Formulars ergibt sich aus der Dialog-ID **DIALOG_<dialogID>**. Derzeit bestehen solche Formulare aus exakt einem Element **<page>** mit einem einzigen Element **<rowGroup>**. 5. Fügen Sie der **rowGroup** für jedes Attribut ein Element **<row>** mit einem Element **<element>** hinzu. Die XML-Attribute **id** (**row**) und **attribute.idref** (**element**) verweisen auf die entsprechende ID des Dialogattributs. 6. Passen Sie die entsprechende **commandchain** des Objekts an (siehe vorheriges Kapitel).
Dokumente	dialogs.xsd, forms.xsd, dialogs.xml, forms_*.xml
Beispiel	AddNewInputDialog \WEB-INF\config\custom\xml\custom.xml: Add custom dialog

4.3 Stammdatenimport anpassen

Um ein Attribut aus ARIS Architect einem Attribut in ARIS Risk & Compliance Manager zuzuordnen, muss die Mapping-Datei für den ARIS Architect-Export-Report entsprechend angepasst werden. Ein Attribut-Mapping besteht immer aus dem Namen in ARIS Risk & Compliance Manager, dem Attributtyp aus ARIS Risk & Compliance Manager (siehe objectTypes.xml) und dem Attributtyp aus ARIS Architect. Dieser Attributtyp kann einen direkten Attributtyp aus ARIS Architect benennen oder ein besonderes Schlüsselwort sein. Das Customizing bietet die folgenden Möglichkeiten:

- **<ABA constant name>**: Konstante eines Standardattributs aus der Methode von ARIS Architect, z. B. AT_NAME.
- **<ABA attribute GUID>**: Wenn das Attribut kein Standardattribut von ARIS Architect ist, können Sie diese GUID verwenden.
- **FALSE**: Entspricht CONSTANT#false
- **TRUE**: Entspricht CONSTANT#true
- **DATE_NOW**: Gibt die Ausführungszeit des Reporte als Wert wieder.
- **ISMULTIPLE**: Für den Export von multiEnum-Attributen. Es werden alle Attribute, die im jeweiligen Enum-Mapping aufgelistet sind, durchsucht. Die Werte werden gelesen und durch Kommas getrennt ausgegeben.
- **CONSTANT#<Constant value>**: Gibt den spezifizierten konstanten Wert aus.

Eine reine Anpassung der Mapping-Datei ist nur für die Attribute von ARIS Risk & Compliance Manager ausreichend die keine List-Attribute sind, da diese die Attributwerte aus ARIS Architect erhalten. Abgesehen von der Anpassung der Mapping-Datei, verlangt das Mapping der List-Attribute auch ein Umschreiben der Reporte. Bei einer späteren Ausführung des Export-Reports, muss sichergestellt werden, dass die hinzugefügten Attribute von ARIS Architect im aktuellen Filter von ARIS Architect enthalten sind, da die Werte ansonsten nicht ausgelesen werden können.

Speicherort	XML-Datei **aris2arcm_mapping_RBA.xml** oder **aris2arcm_mapping_CBA.xml** in ARIS Architect
Dokumente	aris2arcm_mapping_RBA.xml, aris2arcm_mapping_CBA.xml
Beispiel	▪ ModifyMapping_ABA\WEB-INF\config\custom\xml\custom_mapping_string.xml: Add string attribute ▪ ModifyMapping_ABA\WEB-INF\config\custom\xml\custom_mapping_constant.xml: Add constant number attribute ▪ ModifyMapping_ABA\WEB-INF\config\custom\xml\custom_mapping_single_enum.xml: Add single enum attribute ▪ ModifyMapping_ABA\WEB-INF\config\custom\xml\custom_mapping_multi_enum.xml: Add multi enum attribute

4.4 Hierarchien hinzufügen / anpassen

Im Folgenden wird beschrieben, wie ein neuer Hierarchietyp hinzugefügt werden kann und wie über die neue Hierarchie eine einfache Auswertung hinzugefügt wird.

4.4.1 Enumeration-Item hinzufügen

In ARIS Risk & Compliance Manager wird eine neue Hierarchie eingefügt. In den Propertys wird ein Name für die Hierarchie vergeben. Danach muss ein neuer Mandant angelegt werden, damit die Änderung sichtbar wird, denn das initiale Wurzelelement der Hierarchie wird erst beim Generieren des Mandanten in die Datenbank generiert.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Fügen Sie ein neues Element **<enumitem>** an der gewünschten Stelle hinzu. Die Eigenschaft **id** erhält einen für die Hierarchie eindeutigen Namen, die Eigenschaft **value** enthält einen numerisch eindeutigen Identifizierer, der in die Datenbank übertragen wird.
Dokumente	enumerations.xml, enumerations.xsd
Beispiel	AddHierarchy \WEB-INF\config\custom\xml\custom.xml: Step 1

Speicherort	Property-Datei im Ordner **properties**
Vorgehen	Fügen Sie eine neue Property an der gewünschten Stelle hinzu und geben Sie ihr den gewünschten Hierarchienamen.
Beispiel	AddHierarchy \WEB-INF\config\custom\properties\application\custom.properties

4.4.2 Neues Listenelement in Stammdatenobjekt hinzufügen

Damit die Hierarchie in der Anwendung ausgewertet werden kann, muss sie einem Element von ARIS Risk & Compliance Manager zugewiesen werden, das in einer logischen Beziehung zu dieser neuen Hierarchie steht. Das Hinzufügen neuer Elemente ist im Kapitel **Einfaches Attribut hinzufügen / ändern** (Seite 23) beschrieben.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Fügen Sie ein neues Element **<listAttrType>** an der gewünschten Stelle hinzu. In den neuen listAttrType fügen Sie ein Element **<listRestriction>** mit dem inneren Element **<attributeRestriction>** ein. Hier müssen Sie den Hierarchietyp einschränken, der an das neue Attribut angefügt werden kann.
Dokumente	objectTypes.xml, objectTypes.xsd
Beispiel	AddHierarchy \WEB-INF\config\custom\xml\custom.xml: Step 2

Speicherort	Property-Datei im Ordner **properties**
Vorgehen	Fügen Sie eine neue Property an der gewünschten Stelle hinzu und geben Sie ihr den gewünschten Namen des neuen List-Attributs.
Beispiel	AddHierarchy \WEB-INF\config\custom\properties\application\custom.properties

4.4.3 Neues Listenelement in Transaktionsobjekt hinzufügen

Die Beziehung zu der neuen Hierarchie soll nun auf das transaktionale Objekt übertragen werden mit dem die Objekt-Owner arbeiten und das einen Status besitzt, der über die Hierarchie ausgewertet werden kann. Das Hinzufügen neuer Listenelemente ist im Kapitel **Einfaches Attribut hinzufügen/ändern** (Seite 23) beschrieben.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Fügen Sie ein neues Element **<listAttrType>** an der gewünschten Stelle hinzu. In den neuen listAttrType fügen Sie ein Element **<listRestriction>** mit dem inneren Element **<attributeRestriction>** ein. Hier schränken Sie den Hierarchietyp ein, der in das neue Attribut eingefügt werden kann.
Dokumente	objectTypes.xml, objectTypes.xsd
Beispiel	AddHierarchy \WEB-INF\config\custom\xml\custom.xml: Step 3

Speicherort	Property-Datei im Ordner **properties**
Vorgehen	Fügen Sie eine neue Property an der gewünschten Stelle hinzu und versehen Sie sie mit dem gewünschten Namen des neuen Listenattributs.
Beispiel	AddHierarchy \WEB-INF\config\custom\properties\application\custom.properties

4.4.4 Eingabemöglichkeit und Anzeige in den Formularen

Es besteht die Möglichkeit, die Daten der neuen Hierarchie aus ARIS Architect zu importieren, da hier normalerweise auch die Struktur gepflegt wird. In unserem Beispiel schaffen wir allerdings selbst eine Eingabemöglichkeit in dem Formular des Stammdatenobjekts. Das Hinzufügen neuer Listenelemente in Formulare ist unter **Attribut zu einem Formular hinzufügen (Seite 25)** beschrieben.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Fügen Sie ein neues Element **<row>** an der gewünschten Stelle hinzu. In der neuen Row fügen Sie ein Element **<element>** mit dem inneren Element **<parameter>** ein. Hier stellen Sie die Selection-List auf **AUTO**. Dadurch wird die Selektion eines neuen Hierarchieelements möglich ohne dass Sie eine zusätzliche Liste zum Selektieren schreiben müssen.
Dokumente	forms_testmanagement.xml, forms.xsd
Beispiel	AddHierarchy \WEB-INF\config\custom\xml\custom.xml: Schritt 4 und Schritt 5

Daneben ist es natürlich auch möglich, dieses neue Attribut in Listen anzuzeigen. Siehe hierzu **Attribut zu einer Liste hinzufügen** (Seite 27).

4.4.5 Automatische Übertragung von Hierarchie-Objekten

Entsprechen die Namen der Attribute den Konventionen, d. h. sie stimmen im Stammdaten- und transaktionalen Objekt überein, wird der Generator (im Beispiel beim Generieren von Testfällen) das Hierarchieobjekt automatisch übertragen. Die Konventionen sind im Abschnitt **Konventionen bei der Objektgenerierung** (Seite 12) beschrieben.

4.4.6 Hierarchie-Attribut bearbeitbar machen

Das Bearbeiten der Regeln wird im Abschnitt **Regel hinzufügen/anpassen (Seite 77)** beschrieben.

Speicherort	DRL-Datei im Ordner **rules**.
Vorgehen	Hier wird in der Regel **Define all standard attributes editable** das Attribut im Stammdatenobjekt bearbeitbar gemacht.
Dokumente	risk.drl
Beispiel	AddHierarchy \WEB-INF\config\custom\rules\risk.drl: Enable the editing of a hierarchy attribute in rules

4.4.7 Rollen zu Hierarchie-Attribut zuweisen

Sie können ein Hierarchieelement zu einer Rolle hinzufügen oder entfernen. Das Bearbeiten von Rollen ist im Abschnitt **Rollen anpassen (zuweisen/entfernen) (Seite 34)** beschrieben.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Verknüpfen Sie im Element **<relation>** das Listenattribut mit einem Assign- und/oder Remove-Recht.
Dokumente	roles.xsd
Beispiel	AddHierarchy \WEB-INF\config\custom\xml\custom.xml: Step 6

4.4.8 Hierarchieauswertung hinzufügen

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Fügen Sie ein neues Element **<nav.evaluation>** hinzu und verknüpfen Sie es mit der bereits existierenden Statistikdefinition.
Dokumente	navigation_evaluation.xml, navigations.xsd, evaluations.xml, evaluations.xsd
Beispiel	AddHierarchy \WEB-INF\config\custom\xml\custom.xml: Step 9

4.4.9 Neue Daten-View für Hierarchiestatistik anlegen

Die Statistik muss mit einer neuen Daten-View ausgestattet werden. Das Bearbeiten von Statistik-Views ist im Abschnitt **Statistik anpassen** (Seite 62) erläutert.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Legen Sie im Element **<view>** eine neue Daten-View an, die die neue Hierarchie mit dem transaktionalen, auszuwertenden Objekt und der User-Gruppe verknüpft.
Dokumente	views.xsd
Beispiel	AddHierarchy \WEB-INF\config\custom\xml\custom.xml: Step 7

Ebenso muss eine neue View, die aus den Statistiken die verlinkten Listen der verbundenen transaktionalen Objekte generiert, geschrieben werden.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Legen Sie im Element **<view>** eine neue Daten-View an, die aus den Statistiken die verlinkten Listen der verbundenen transaktionalen Objekte generiert.
Dokumente	views.xsd
Beispiel	AddHierarchy \WEB-INF\config\custom\xml\custom.xml: Step 8

4.5 Statistiken hinzufügen / anpassen

Als Statistiken werden Auswertungen bezeichnet, die eine Verteilung von Daten über eine Struktur anzeigen. Die Testfall-Statistik zeigt z. B. die Verteilung der Testergebnisse über eine Hierarchie.

4.5.1 Statistik anpassen

Bei Statistiken können Sie mit Hilfe der XML-Konfiguration folgende Aktionen durchführen (sortiert nach Komplexität):

- Spaltenbreite anpassen
- Strukturelemente auf zugehörige Formulare verlinken
- Spalte(n) hinzufügen/anpassen
  - statistic.columnGroup.enum-basierter Header
  - statistic.columnGroup.perCent-basierter Header
  - statistic.column.value-basierter Header
- Verlinkungen anpassen
- Neue Hierarchie einbinden

4.5.1.1 Spaltenbreiten anpassen

Das Width-Attribut ist für alle Spalten optional. Wenn keine Spaltenbreiten konfiguriert werden, wird die Breite der einzelnen Spalten automatisch berechnet. Standardmäßig wird der Struktur 20 % und jeder Chart-Spalte eine Breite von 1 % zugewiesen. Der Rest wird gleichmäßig auf die übrigen Spalten verteilt.

Da im Excel- und PDF-Report keine Chart-Spalten enthalten sind, wird das Layout für einen Report neu berechnet, d. h., die Breite der Chart-Spalten wird auf die im Report sichtbaren verteilt.

4.5.1.2 Strukturelemente verlinken

Der Strukturbaum der Statistik enthält in der Regel Knoten, die Objekte repräsentieren, z. B. Hierarchien oder Benutzergruppen. Diese Objekte können mit dem Knoten des Strukturbaums verlinkt werden. Die verlinkten Objekte werden als Popup-Fenster angezeigt.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Kopieren Sie das Element **<evaluation>** aus dem Standard in das Customizing. Danach können Sie ein Element **<statistic.column.tree>** um das Attribut **linkedNodeTypes** erweitern. Als Attributwert muss eine komma-separierten Liste der IDs der Objekttypen vorliegen.
Bemerkung	Ein Strukturelement, welches einen Objekttyp der komma-separierten Liste repräsentiert, wird als Link dargestellt. Der Link öffnet das zugehörige Objekt in einem Popup-Fenster
Dokumente	evaluations.xsd, evaluations_*.xml
Beispiel	ModifyStatistic \WEB-INF\config\custom\xml\evaluation_custom.xml: Add custom linked nodes

4.5.1.3 Spalten hinzufügen / anpassen

4.5.1.3.1 statistic.columnGroup.enum-basierte Statistik

Auswertungen, die eine prozentuale Verteilung von Enumeration-Attributwerten von bestimmten Objekten anzeigen, werden automatisch um eine Spalte erweitert, wenn die entsprechende Enumeration erweitert wird.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Kopieren Sie das Element **<enum>** aus dem Standard in das Customizing. Danach können Sie ein oder mehrere neue **<enumitem>** innerhalb des Elements **<enumitem>** anlegen. Innerhalb des Elements **<enumitem>** können Sie mittels des Elements **<parameter>** die Farbe des Spaltenkopfs angeben. Diese Farbe wird auch in den Tortendiagrammen zur Anzeige verwendet. **Beispiel** `<parameter name="Background" value="EBB585"/>`
Bemerkung	Gilt nur für Statistiken, deren Spalten mit dem XML-Element **<statistic.columnGroup.enum>** konfiguriert sind. Virtuelle Enumeration-Items werden im Spaltenkopf zur Gliederung verwendet. D. h. reale Items sind den virtuellen Items im Header untergeordnet. Die Anzeige von Spalten, die Enumeration-Attributwerte (real oder virtuell) repräsentieren, kann durch Einfügen des Attributs **evaluationRelevant** mit dem Wert **false** verhindert werden.
Dokumente	evaluations.xsd, evaluations_*.xml
Beispiel	ModifyObjectLifecycle \WEB-INF\config\custom\xml\testcase_enum_custom.xml: New item as a configuration example

4.5.1.3.2 statistic.columnGroup.perCent-basierte Statistik

Dieses Element wird verwendet, um Absolut- und Prozent-Werte für Nicht-Enumeration-Werte anzuzeigen.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Kopieren Sie das Element **<evaluation>** aus dem Standard in das Customizing. Danach können Sie ein Element **<statistic.columnGroup.perCent>** anpassen. Die einzelnen Spalten der Gruppe können Sie durch Elemente **<statistic.column.value>** hinzufügen. Wenn Sie zur Berechnung **<statistic.calculator id="value" />** verwenden, können Sie mit dem Wert des Attributs **calculation** die Berechnung der Spaltenwerte steuern.
Bemerkung	Mögliche Werte des Attributs **calculation** sind: **count**: Für jeden gefundenen Wert in der Datenquelle wird der Wert der Spalte um **1** erhöht. **sum**: Alle Werte der Datenquelle werden addiert. Die Werte der Datenquelle müssen vom Typ **Number** sein. Jede Spaltengruppe beinhaltet standardmäßig eine Summenspalte und ein Tortendiagramm. Die Summenspalte enthält die Summe der Werte aller per Element **<statistic.column.value>** konfigurierten Spalten. Das Tortendiagramm enthält ein Stück pro konfigurierte Spalte.
Dokumente	evaluations.xsd, evaluations_*.xml
Beispiel	ModifyStatistic \WEB-INF\config\custom\xml\evaluation_custom.xml: Add custom value column perCent

4.5.1.3.3 statistic.column.value-basierte Statistik

Einzelne Werte, die nicht zu einer Gruppe von Prozentwerten gehören, können auch mit dem Element **<statistic.column.value>** hinzugefügt und angepasst werden. Der Wert wird dann als absoluter Wert angezeigt.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Kopieren Sie das Element **<evaluation>** aus dem Standard in das Customizing. Danach können Sie ein Element **<statistic.column.value>** anpassen. Wenn Sie zur Berechnung **<statistic.calculator id="value" />** verwenden, kann mit dem Wert des Attributs **calculation** die Berechnung der Spaltenwerte gesteuert werden.
Bemerkung	Mögliche Werte des Attributs **calculation** sind: **count**: Für jeden gefundenen Wert in der Datenquelle wird der Wert der Spalte um **1** erhöht. **sum**: Alle Werte der Datenquelle werden addiert. Die Werte der Datenquelle müssen vom Typ **Number** sein.
Dokumente	evaluations.xsd, evaluations_*.xml
Beispiel	ModifyStatistic \WEB-INF\config\custom\xml\evaluation_custom.xml: Add custom value column

4.5.1.3.4 Verlinkungen anpassen

Jede Zelle bzw. die darin angezeigten Werte können verlinkt werden. In der Regel repräsentiert die Verlinkung den Wert der Zelle. Beispiel: Der Link einer Zelle mit der Zahl 10 in einer Spalte, die offene Testfälle anzeigt, öffnet eine Liste mit 10 offenen Testfällen.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Kopieren Sie das Element **<evaluation>** aus dem Standard in das Customizing. Danach können Sie die Attribute **link.typ** und **link.idref** der Elemente **<statistic.column.value>**, **<statistic.columnGroup.perCent>** und **<statistic.columnGroup.enum>** anpassen oder hinzufügen. Als link.typ-Attributwert sind **list** und **evaluation** erlaubt. Als link.idref-Attributwert wird eine ID aus der Datei **lists_*.xml** bzw. **evaluations_*.xml** erwartet.
Bemerkung	Die verlinkte Liste bzw. Auswertung wird mit den gleichen Werten gefiltert mit denen auch die Datenquelle der Statistik gefiltert wurde. D. h. die View der Liste bzw. der verlinkten Auswertung unterliegt den gleichen Bedingungen.
Dokumente	evaluations.xsd, evaluations_*.xml, lists_*.xml
Beispiel	ModifyStatistic \WEB-INF\config\custom\xml\evaluation_custom.xml: Add custom linked column

4.5.1.3.5 Neue Hierarchie verwenden

Details zur Verwendung neuer Hierarchien finden Sie im Kapitel **Hierarchien hinzufügen / anpassen** (Seite 56).

4.6 Reporte hinzufügen / anpassen

Reportdefinitionen können in sehr unterschiedlicher Weise angepasst werden. Reportanpassungen allein durch Änderungen an der XML-Konfiguration sind nur für Formular- und Listenreporte möglich. Sie werden in den beiden folgenden Kapiteln beschrieben.

4.6.1 Reporte für Formulare hinzufügen / anpassen

Für einige Formulare z. B. den Fragebogen gibt es bereits spezialisierte Reportdefinitionen. Andere Objekte wie Risiko und Kontrolle hingegen besitzen diese nicht. In diesem Fall wird automatisch vor der Ausführung eine Reportdefinition aus dem Formular generiert und direkt angewendet. Für Formulare und Listen werden diese Reportdefinitionen in die Datei **reports_dynamicreports.xml** geschrieben, um als Customizing-Beispiel dienen zu können. Die folgenden Beispiele zeigen, wie generell bestehende Reportdefinitionen für Formulare ersetzt werden können und wie neue Reportdefinitionen für solche Formulare festgelegt werden können, die noch keine spezialisierten Reportdefinitionen besitzen.

4.6.1.1 Bestehende Formularreportdefinition ersetzen

Um einen bestehenden Formularreport zu ersetzen, genügt es, in einer beliebigen Customizing-XML eine Reportdefinition mit identischer ID und Format anzugeben. Diese Reportdefinition wird anstelle der Standarddefinition verwendet. Das folgende Beispiel ersetzt den Standard-Formularreport der Befragung durch eine verkürzte Version, die die Ausgabe der Fragen auslässt.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Kopieren Sie die Reportdefinition des Formulars und verändern Sie sie dann beliebig.
Dokumente	Report-XML-Dateien im Standard als Vorlage.
Beispiel	FormReports_ReplaceExisting \WEB-INF\config\custom\xml\custom.xml: Replace standard questionnaire form report with a simplified version

4.6.1.2 Neue Formularreportdefinition hinzufügen

Die Vorgehensweise, einen Formularreport für die Formulare hinzuzufügen, die keine spezialisierte Reportdefinition besitzen, ist identisch mit dem vorherigen Beispiel. Es ist aber erforderlich, dass die Report-ID dem Formular-Objekttyp in Großbuchstaben entspricht. Das folgende Beispiel definiert einen PDF-Formularreport für Risiken. Die Report-ID ist daher **RISK**. Sie verweist auf den untergeordneten Formularreport **CONTROL**. Da dieser weiterhin nicht explizit definiert wird, verwendet ARIS Risk & Compliance Manager hier nach wie vor eine direkt aus dem Formular generierte Definition.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Legen Sie eine neue Reportdefinition an.
Dokumente	Report-XML-Dateien im Standard als Vorlage.
Beispiel	FormReports_AddNew \WEB-INF\config\custom\xml\custom.xml: Add a new risk form report

4.6.1.3 Neue Formularreportauswahl einbinden

Anstelle eines Formularreports kann ein Auswahldialog konfiguriert werden, der auf mehrere andere Reporte verweist. Die ID des Auswahldialogs muss entweder genau dem Formularobjekttyp in Großbuchstaben entsprechen oder das zusätzliche Suffix **_SELECT** haben. Das folgende Beispiel definiert eine Reportauswahl für den Excel-Formularreport einer Risikobewertung, bei der zusätzlich zwischen dem Standardformularreport, einem speziell definierten Report und einer Kombination der beiden gewählt wird. Der Standard-Form-Report wird für das Excel-Format automatisch generiert.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Legen Sie eine neue Reportdefinition an und aktivieren Sie **Excel report** im Formular.
Dokumente	Report-XML-Dateien im Standard als Vorlage.
Beispiel	▪ 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

4.6.2 Reporte für Listen hinzufügen / anpassen

Für einige Listen wie beispielsweise die Benutzerliste oder die Testfalllisten gibt es bereits spezialisierte Reportdefinitionen. Andere Listen, wie die Risiko- oder Kontroll-Liste hingegen, besitzen diese nicht. In diesem Fall wird automatisch vor der Ausführung eine Reportdefinition aus der jeweiligen Liste generiert und direkt angewendet. Für Formulare und Listen werden diese Reportdefinitionen in die Datei **reports_dynamicreports.xml** geschrieben, um als Customizing-Beispiel dienen zu können. Die folgenden Beispiele zeigen, wie generell bestehende Reportdefinionen für Listen ersetzt werden können, und wie neue Reportdefinitionen für solche Listen festgelegt werden können, die noch keine spezialisierten Reportdefinitionen besitzen.

4.6.2.1 Bestehende Listenreportdefinition ersetzen

Um einen bestehenden Listenreport zu ersetzen, genügt es, in einer beliebigen Customizing-XML eine Reportdefinition mit derselben ID und demselben Format anzugeben. Diese Reportdefinition wird anstelle der Standarddefinition verwendet. Das folgende Beispiel ersetzt den Standard-Listenreport für Benutzer durch eine verkürzte Version, die die Spalten reduziert.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Kopieren Sie die Reportdefinition der Liste und passen Sie diese beliebig an.
Dokumente	Report-XML-Dateien im Standard als Vorlage.
Beispiel	ListReports_ReplaceExisting \WEB-INF\config\custom\xml\custom.xml: Replace standard user list report with a simplified version

4.6.2.2 Neue Listenreportdefinition hinzufügen

Die Vorgehensweise, einen Listenreport für die Listen hinzuzufügen, die keine spezialisierte Reportdefinition besitzen, ist identisch mit dem vorigen Beispiel. Die Report-ID muss dabei mit der Listen-ID identisch sein. Das folgende Beispiel definiert einen PDF-Listenreport für die Risiken-Liste im Explorer, die für den Test-Manager angezeigt wird. Die Report-ID ist daher **RISK**.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Legen Sie eine neue Reportdefinition an.
Dokumente	Report-XML-Dateien im Standard als Vorlage.
Beispiel	ListReports_AddNew \WEB-INF\config\custom\xml\custom.xml: Add a new risk list report

4.6.2.3 Neue Reportauswahl einbinden

Anstelle eines Listenreports kann ein Auswahldialog konfiguriert werden, der auf mehrere andere Reporte verweist. Die ID des Auswahldialogs muss entweder genau der Listen-ID entsprechen oder das zusätzliche Suffix _**SELECT** haben. Das folgende Beispiel definiert eine Reportauswahl für den Excel-Listenreport der Risikobewertungsliste im Explorer, bei der zusätzlich zwischen dem Standard-Listen-Report, einem speziell definierten Report und einer Kombination der beiden gewählt wird. Der Standard-Listen-Report wird für das Excel-Format automatisch generiert.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Legen Sie eine neue Reportdefinition an.
Dokumente	Report-XML-Dateien im Standard als Vorlage.
Beispiel	▪ 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

4.7 Modify message template

4.7.1 Neues Message-Template hinzufügen

Um ein neues Message-Template verwenden zu können, muss dieses zuerst hinzugefügt werden.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	1. Kopieren Sie das Element **<enum>** mit der ID **initiators** aus dem Standard in das Customizing. 2. Legen Sie neue Message-Templates mit Hilfe des Elements **<enumitem>** innerhalb des Elements **<enum>** an. Die Attribute **id** und **value** müssen mindestens gesetzt werden. Der Wert für beide Attribute muss dabei innerhalb des Elements **<enum>** eindeutig sein. 3. Legen Sie für das neu angelegte Message-Template einen neuen Property-Eintrag an (siehe **Propertys für Enumeration hinzufügen/anpassen** (Seite 31)). Dieser wird dazu verwendet, das Message-Template in der Vorlagenliste anzuzeigen.
Dokumente	enumerations.xml, enumerations.xsd
Beispiel	▪ 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

4.7.2 Neuem Message-Template Inhalt hinzufügen

Nachdem ein neues Message-Template hinzugefügt wurde, muss danach der Inhalt für dieses neue Message-Template hinzugefügt werden.

Vorgehen	Fügen Sie zwei neue Property-Einträge in der custom.properties-Datei hinzu (siehe **Propertys hinzufügen/anpassen** (Seite 23)).
Bemerkung	Die Namen der beiden neuen Einträge müssen folgende Konventionen erfüllen: Subject of the message: message.<template_name>.subject.DBI Contents of the message: message.<template_name>.text.DBI
Beispiel	ModifyMessageTemplate_AddNewMessageTemplate \WEB-INF\config\custom\properties\application\custom.properties: Add contents to new message template

4.7.3 Inhalt des Message-Templates anpassen

Der Inhalt eines Message-Templates kann direkt in ARIS Risk & Compliance Manager angepasst werden.

Vorgehen	1. Klicken Sie auf **Administration**. 2. Klicken Sie unter **Systemmanagement** auf **Mandanten** bzw. auf **System**, je nachdem welche Nachrichtenvorlage Sie bearbeiten möchten. 3. Öffnen Sie die Nachrichtenvorlage, die Sie anzeigen lassen oder ändern möchten. Der entsprechende Text wird im Feld **Nachricht** angezeigt. 4. Ändern Sie ggf. den Text der Vorlage bzw. den Betreff. 5. Wenn Sie den Text aus einem anderssprachigen ARIS Risk & Compliance Manager versenden möchten, wählen Sie im Feld **Sprache** die gewünschte Sprache und geben Sie den Nachrichtentext im Feld **Vorlage** ein. 6. Entscheiden Sie im Feld **Benachrichtigungsart**, ob die Benachrichtigungen per E-Mail und/oder an die interne Mailbox von ARIS Risk & Compliance Manager versandt werden sollen. Nachdem Sie die Änderung gespeichert haben, wird der Inhalt des Message-Templates in der Datenbank gespeichert. Solange das Message-Template nicht wieder über die Oberfläche zurückgesetzt wird, wird der Inhalt aus der Datenbank verwendet. Sie haben außerdem die Möglichkeit ein Message-Template zu ändern, indem Sie wie beim Hinzufügen eines neuen Message-Templates vorgehen (siehe **Add new message template** (Seite 72)). In diesem Fall wird allerdings der Property-Wert einer bestehenden Vorlage angepasst.
Bemerkung	Werden in einem Message-Template Werte benötigt, die erst zur Laufzeit dynamisch ermittelt werden können, werden diese in Form von Objekten und Variablen integriert. Der Unterschied zwischen diesen beiden ist, dass ein Objekt entsprechende Methoden zum Zugriff auf Werte bereitstellt und eine Variable einen festen Wert besitzt. Standardmäßig steht immer das jeweilige Objekt, das für den Formularfluss verantwortlich ist, als Objekt zur Verfügung. Auf alle für diesen Objekttyp definierten Attributen, inklusive **Vererbung** (Seite 4) kann mittels der entsprechenden Methode zugegriffen werden. Konventionen zur Verwendung von Objekten in einem Message-Template: ▪ Das Zugreifen auf ein Objekt oder eine Variable muss immer mit dem Präfix **$** erfolgen. ▪ Der Name des Objekts entspricht immer der definierten Attribut-ID des Elements **<objectType>** in Kleinbuchstaben.

	▪ Der Name der Methode setzt sich immer aus dem Präfix **get** und der Attribut-ID des Elements **<attrType>** zusammen. Zu beachten ist, dass nach dem **get** immer ein Großbuchstabe folgen muss. Des Weiteren werden Unterstriche in der ID heraus gefiltert und durch den Großbuchstaben des folgenden Wortes ersetzt. **Beispiel** Objekttyp **Testdefinition** mit definierter ID **TESTDEFINITION** und dem Attribut **owner_group** Zugriff auf dieses Attribut in Message-Template: **$testdefinition.getOwnerGroup()** Zusätzliche Objekte und Variablen, die standardmäßig zur Verfügung stehen: ▪ **$user** – Objekt mit Empfängerinformationen ▪ **$client** – Objekt mit Mandanteninformationen ▪ **$serverConnection** – Variable mit Link zum Server Sollten zusätzliche Objekte oder Variablen benötigt werden, funktioniert dies nur durch Implementierung von Java-Code. Ein Indiz für eigene Implementierungen sind Message-Templates, in denen zusätzliche Objekte und Variablen bereits verwendet werden.
Dokumente	objectTypes.xml, objectTypes.xsd
Beispiel	ModifyMessageTemplate_AddNewMessageTemplate \WEB-INF\config\custom\properties\application\custom.properties: Add contents to new message template

4.7.4 Nachrichten versenden

Nachdem ein neues Message-Template angelegt wurde, kann dieses in den Formularfluss eingebunden und versendet werden.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	1. Kopieren Sie ein Element **<catalog>** aus dem Standard in das Customizing. 2. Fügen Sie durch das Element **<command> sendMail** das neu angelegte Message-Template innerhalb des Elements **<commandchain>** hinzu. 3. Legen Sie mithilfe der Elemente **<parameter>** das neue Message-Template sowie die Empfänger der Nachricht fest. Der Parameter **cc** ist dabei optional und muss nicht verwendet werden. Erlaubter Wert für die Parameter **to** und **cc** ist immer das definierte Attribut **id** des zugehörigen Elements **<listAttrType>**, welches zusätzlich als Attribut **objectType.idref** die Werte **USER** oder **USERGROUP** enthält.
Dokumente	commandchains_[module].xml, commandchains.xsd, objectTypes.xml, objectTypes.xsd
Beispiel	ModifyMessageTemplate_AddNewMessageTemplate \WEB-INF\config\custom\xml\custom.xml: Add send mail command to send new message

4.8 Aufgabentrennung hinzufügen / anpassen

Die Aufgabentrennung wird verwendet, um für bestimmte Objekte das Vier-Augen-Prinzip zu erzwingen. Dies wird durch die Konfiguration von Rollenpaaren erreicht, die ein Benutzer nicht haben darf, um ein solches Objekt bearbeiten zu können. So darf zum Beispiel ein Testfall nicht von einem Benutzer bearbeitet werden, der am Testfall die Tester- und Reviewer-Rolle hat.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	1. Kopieren Sie das Element **<segregationsOfDuties>** aus dem Standard in das Customizing. 2. Legen Sie ein oder mehrere neue Elemente **<segregationsOfDuties>** innerhalb des Elements **<segregationsOfDuties>** an. 3. Geben Sie den Objekttyp des Objekts, für das eine Aufgabentrennung konfiguriert werden soll, im Attribut **objectType** an. 4. Geben Sie innerhalb des Elements **<segregationsOfDuties>** in genau zwei Elementen **<segregationsOfDuties>** die beiden Rollen an, die ein Benutzer nicht haben darf, um ein Objekt zu bearbeiten.
Bemerkung	Sie können ein Element **<segregationsOfDuties>** auch entfernen, um das Vier-Augen-Prinzip aufzuheben.
Dokumente	segregationsOfDuties.xsd, segregationsOfDuties.xml
Beispiel	<segregationOfDuties objectType="testcase"> <segregationsOfDuties.role id="tester"/> <segregationsOfDuties.role id="testreviewer"/> </segregationOfDuties>

4.9 Regel hinzufügen / anpassen

Regeln werden eingesetzt, um das Verhalten von Formularen zu manipulieren. Die Regeln, die für ein jeweiliges Formular angewendet werden sollen, sind in der Standardkonfiguration in jeweils einer DRL-Datei zusammengefasst, die den Namen des Objekttyps aus dem Formular trägt (beispielsweise risk.drl).

4.9.1 Bestehende Regel-Datei überschreiben

Um eine bestehende Regeldatei zu überschreiben, muss eine gleichnamige Datei unter dem Pfad **WEB-INF / config / custom / rules** abgelegt werden. Beim Start des Servers wird diese Datei anstelle der Standard-Datei gelesen. Sämtliche definierte Conditions und Consequences der Regeln sind sprachliche Beschreibungen der Funktionalität, die von der Regel implementiert werden soll. Sie müssen definierte DSL-Items sein, die in der DSL-Datei aufgeführt sind, welche wiederum in der DRL-Datei selbst sowie in ihrem definierten Rule-Set referenziert wird. Dieses Beispiel zeigt, wie Sie für das Objekt **USERGROUP** ein neues Attribut hinzufügen, das Formular entsprechend erweitern und die bestehenden Formularregeln durch eine erweiterte Variante ersetzen. Es umfasst auch die Definition eines neuen Attributs und die Anpassung des Formulars.

Speicherort	DRL-Datei im Ordner **rules**.
Vorgehen	Kopieren Sie die DRL-Datei des gewünschten Formulars in den oben genannten Ordner. Ändern Sie gegebenenfalls bestehende Regeln oder ergänzen Sie die Datei um neue Regeln.
Dokumente	Entsprechende DSL-Datei im Standard Java-Doc der Klasse **CollectiveHelper** und deren abgeleitete Klassen zur Übersicht der möglichen Conditions und Consequences.
Beispiel	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'

4.9.2 Neue Regel-Datei einbinden

Um eine neue Regel-Datei einzubinden, muss ein neues Rule-Set definiert werden, das auf dieselbe DSL-Datei verweist, die auch innerhalb der Regeldatei referenziert wird. Dieses Rule-Set muss in einer Custom-XML im Pfad **WEB-INF/config/custom/xml** abgelegt werden. Ebenso muss der bestehende Rule-Context, zu dem das neue Rule-Set gehören soll, überschrieben und ebenso in einer Custom-XML im obigen Pfad abgelegt werden. Das folgende Beispiel ergänzt die Regeln des USERGROUP-Formulars um zwei weitere Regeln.

Speicherort	DRL-Datei im Ordner **rules** XML-Datei im Ordner **xml**
Vorgehen	1. Kopieren Sie die DRL-Datei des gewünschten Formulars. 2. Behalten Sie den Header bis zu den Import-Statements bei und löschen Sie den restlichen Inhalt. 3. Fügen Sie nach Bedarf neue Regeln ein. 4. Binden Sei die neue DRL-Datei in ein neues ruleSet ein und speichern Sie dieses in einer custom.xml-Datei. 5. Überschreiben Sie den ruleContext des entsprechenden Formulars und binden Sie das neue Rule-Set neben dem Standard-Set ein.
Dokumente	Entsprechende DRL-Datei im Standard Entsprechende DSL-Datei im Standard rulesetReg.xml im Standard Java-Doc der Klasse **CollectiveHelper** und deren abgeleitete Klassen zur Übersicht der möglichen Conditions und Consequences.
Beispiel	ModifyRules_AddDRL\WEB-INF\config\custom\rules\usergroup.drl: Add custom rule set ModifyRules_AddDRL\WEB-INF\config\custom\xml\custom_ruleContext.xml: Overwrite the ruleContext and add the custom ruleSet ModifyRules_AddDRL\WEB-INF\config\custom\xml\custom_ruleContext.xml: Add custom ruleSet ModifyRules_AddDRL\WEB-INF\config\custom\xml\custom_usergroupform: Enable rule execution on change of form element 'name'

4.9.3 Bestehende Regeln für neue Attribute wiederverwenden

Um einer bereits bestehende Gruppe von Attributen ein weiteres Attribut hinzuzufügen, das sich hinsichtlich Sichtbarkeit und Pflichtfeld-Bedingungen identisch verhält, kann es am Objekttyp mit dem Merkmal **behavesLike** versehen werden. Das folgende Beispiel zeigt diesen Mechanismus, in dem für den Risiko-Owner ein weiteres Pflichtattribut für den Fall aufgenommen wird, in dem die Risikobewertung mit dem Typ **Qualitative** bewertet werden soll.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	1. Kopieren Sie die Definition des gewünschten Objekttyps und ergänzen Sie das neue Attribut. 2. Geben Sie unter **behavesLike** an diesem Attribut den Namen eines anderen definierten Attributs ein, nach dem sich das Verhalten des neuen Attributs richten soll. 3. Fügen Sie das Attribut im Formular hinzu.
Dokumente	Objecttypes.xml im Standard
Beispiel	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

4.10 Zeitgesteuerte Aufgabe hinzufügen / anpassen

4.10.1 Anpassen der Zeitplanung

ARIS Risk & Compliance Manager bietet prinzipiell zwei Möglichkeiten, automatisierte Jobs zu starten, um Objekte zu generieren oder zu verändern. Die erste Möglichkeit besteht darin, mit einer entsprechenden Managerrolle einen Job manuell über die Oberfläche von ARIS Risk & Compliance Manager anzustoßen. Die zweite Möglichkeit besteht darin, diesen Job zeitgesteuert durch ARIS Risk & Compliance Manager ausführen zu lassen. In der Standardkonfiguration ist das beispielsweise für die Generator-und Überwachungs-Jobs der Fall. Zeitgesteuerte Jobs werden in der Datei **runtimeconfig.xml** eingetragen. Innerhalb des Tags **<section id="scheduler">** besitzt jeder zeitgesteuerte Job eine Parameterliste.

Beispiel

```
<parameterList name="monitorJobTestcase">
          <parameter name="jobitem" value="monitorJob" />
          <parameter name="startScheduler" value="true"/>
          <parameter name="executionTime" value="0 52 01 ? * SUN-SAT"/>
          <parameter name="clientexcludinglist" value=""/>
          <parameter name="clientincludinglist" value=""/>
          <parameter name="objecttypes" value="TESTCASE"/>
</parameterList>
```

Der Name der Parameterliste kann frei gewählt werden, muss jedoch innerhalb der Parameterlisten eindeutig sein. Die einzelnen Parameter haben folgende Bedeutung:

- **Jobitem**

  Der Job, der ausgeführt werden soll. Der Parameterwert muss einer EnumItem-ID aus der Enumeration **jobs** in der Datei **enumerations.xml** entsprechen.

- **startScheduler**

  Muss **true** sein, damit die Zeitsteuerung für diesen Job aktiv ist.

- **executionTime**

  Dieser Ausdruck gibt an, zu welchem Zeitpunkt der Job gestartet werden soll. Er hat das Format **CronTrigger**, welches die Angabe von Zeitintervallen erlaubt.

  Von links nach rechts bedeuten die einzelnen Werte:

  - **Seconds** (0-59)
  - **Minutes** (0-59)
  - **Hours** (0-23)
  - **Day of month** (1-31)
  - **Month** (1-12 oder JAN-DEC)
  - **Day of week** (1-7 oder SUN-SAT)
  - **Year** (kann leer sein, 1984, 1970-2099, ...)

Im obigen Beispiel startet der Monitor-Job zur Überprüfung der Testfälle jeden Tag, in jedem Monat, um 01:52. Weitere Informationen finden Sie in der CronTrigger-Dokumentation auf der Quartz-Homepage (http://www.quartz-scheduler.org (http://www.quartz-scheduler.org))

- **clientexcludinglist**

  Hier werden die Mandanten innerhalb der Datenbank von ARIS Risk & Compliance Manager angegeben, für die der Job nicht ausgeführt werden soll. Die Werte können komma-separiert angegeben werden.

- **Clientincludinglist**

  Hier werden die Mandanten innerhalb der Datenbank von ARIS Risk & Compliance Manager angegeben, für die der Job ausgeführt werden soll. Ist der Wert leer, so wird für jeden einzelnen Mandanten ein separater Job gestartet. Die Werte können komma-separiert angegeben werden.

- **Objecttypes**

  Die Objekttypen, für die der Job ausgeführt werden soll. Im obigen Beispiel soll diese Instanz des Überwachungs-Jobs ausschließlich die Testfälle überprüfen. Die Werte können komma-separiert angegeben werden.

4.10.2 Generator

4.10.2.1 Anpassen der Objektsuche

Die Suche der initialen Recurring-Objekte, ähnlich der Testdefinition für Testfälle, Risiko für Risikobewertungen usw., wird im Standard über die Datei **commandchains_generator.xml** gesteuert. Hier gibt es jeweils eine Chain **generator_[ZielTyp]**, worin entweder das Command **RecurringObjectSearchCommand** oder eine Ableitung davon verlinkt ist.

Dieses Command ist dafür verantwortlich, alle Recurring-Objekte zu finden, die für die Generierung transaktionaler Objekte in Frage kommen. Diese Objekte beziehungsweise deren OVIDs werden vom Command in den CommandChainContext geschrieben und von dort aus durch den Generator ausgelesen und weiterverwendet.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Übernehmen Sie den passenden <catalog> aus der Standardkonfiguration ins Customizing. Danach können Sie die Command-Chain anpassen.
Dokumente	commandClassMapping.xml, blClassMapping.xsd, commandchains_generator.xml, commandchains.xsd

4.10.2.2 Generieren von Objekten

Für jede OVID eines Recurring-Objekts, die der Generator durch die Ausführung der Objektsuche erhält, erzeugt er ein leeres transaktionales Objekt. Ob diesem Objekt korrekt gefüllt und gespeichert werden kann, wird durch die Workflow-Konfiguration des transaktionalen Objekts bestimmt.

Für jeden transaktionalen Objekttyp gibt es im Workflow eine Kante im **<state.initial>**, die nur den Generator-Job als Berechtigten (**permission**) hat. Bei der Erzeugung des Objektes wird der Generator dieser Workflow-Kante folgen.

Beispiel für den Objekttyp TESTCASE

Command-Chain **prepareJob** im Catalog **testcase**.

An dieser Workflow-Kante ist die Chain definiert, mit deren Hilfe das transaktionale Objekt geprüft und gefüllt wird. Der erste Teil der Chain besteht aus Ableitungen der Klasse **GeneratorConditionCheckCommand**, von denen jede für das gegebene Recurring-Objekt, z. B. Testdefinition, prüft, ob alle Bedingungen erfüllt sind. Der zweite Teil besteht aus einer Ableitung des **GenerateCommand**. Hier werden dem neuen transaktionalen Objekt, z. B. Testfall, Attributwerte zugeordnet. Da es sich hier um Commands handelt, kann das Verhalten beider Teile durch Weglassen, Ersetzen und Ergänzen angepasst werden.

Nach der erfolgreichen Abarbeitung der oben beschriebenen Chain befindet sich das transaktionale Objekt im Status **<state.prepared>**. Dies bedeutet, dass der Generator das Objekt zwar erzeugt, aber noch nicht persistiert hat. Im zweiten Schritt folgt der Generator der Kante, für die er berechtigt ist (permission) und die zum ersten Workflow-Status führt, der eine manuelle Bearbeitung durch Benutzer zulässt. Im Standard sind dies die Zustände, an denen Tasks für die Owner oder den Creator definiert sind.

Beispiel für den Objekttyp TESTCASE

Command-Chain **insertJob** im Catalog **testcase**

Innerhalb dieser zweiten Kante ist durch den Befehl **prepareJobMessageCommand** festgelegt, welche Benachrichtigung an die Owner geschickt werden sollen. Der Generator sammelt während seines Durchlaufs die Nachrichten aller erzeugten Objekte, fasst diese so weit wie möglich zusammen und verschickt sie an die oben festgelegten Empfänger.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Übernehmen Sie den passenden <catalog> aus der Standardkonfiguration ins Customizing. Danach können Sie die Command-Chain anpassen.
Dokumente	▪ commandClassMapping.xml, blClassMapping.xsd, ▪ commandchains_[module].xml, ▪ commandchains.xsd, workflow_[module].xml

4.10.3 Überwachungs-Job

4.10.3.1 Anpassen der Objektsuche

Der Überwachungs-Job kann dahingehend angepasst werden, dass transaktionale Objekte, z. B. Testfall, über ihren Workflow dazu bestimmt werden, ob sie vom Überwachungs-Job beachtet werden sollen oder nicht. Wenn ein Objekt in einem bestimmten Workflow-Zustand vom Überwachungs-Job kontrolliert werden soll, muss am Zustand ein Task-Item eingetragen werden. Im Standard sind das immer die Zustände, die die Bearbeitung durch die Owner-Gruppe symbolisieren.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Kopieren Sie den passenden Workflow ins Customizing. Anschließend können Sie die Elemente **<task.item>** an den einzelnen Status verändern.
Bemerkung	Wichtig ist, dass in der Task-Definition die Eigenschaft **time.limitation** nicht auf **false** gesetzt wird, da diese Tasks sonst vom Überwachungs-Job ignoriert werden.
Dokumente	workflow.xsd, workflow_[module].xml

4.10.3.2 Eskalationen

In der Datei **escalations.xml** ist festgelegt, was der Überwachungs-Job mit den gefundenen Objekten macht. Die dort definierten Eskalationen, repräsentiert durch die Elemente **<escalation>**, müssen mit dem Task in Rolle, Name und Objekttyp übereinstimmen. Eine Eskalation umfasst definierte Level, die durch die Elemente **<level>** repräsentiert werden. Der Überwachungs-Job berechnet, wie viel Bearbeitungszeit bereits vergangen ist und welcher Level aktuell erreicht wurde.

Ein bestimmter Level wird erreicht, wenn ein festgelegter Prozentsatz des Bearbeitungszeitraums abgelaufen ist (Attribut **percentage**), oder wenn das Ende der Bearbeitungszeit in einer bestimmten Anzahl von Tagen/Stunden erreicht ist (Attribut **remainingTime**). Wird der nächste Level erreicht, wird dies am Task vermerkt und eine entsprechende Nachricht an die relevante Benutzergruppe verschickt. Die Art der Nachricht, die Gruppe der Empfänger und Attributänderungen an dem Objekt, das vom Task referenziert wird, können ebenfalls angepasst werden:

- Durch Unterelemente **<attributeChange>** kann definiert werden, welche Attribute des referenzierten Objekts welchen Wert erhalten sollen. Die Eigenschaft **id** gibt das Attribut an, welches verändert werden soll. **Value** gibt den neuen Wert für das Attribut an. In der Standardkonfiguration werden z. B. zu 100 % abgelaufene Testfälle auf den Status **nottested** gesetzt.

Attributänderungen sind gleichbedeutend mit Änderungen des Objektes im Formular, d. h. sie können State-Transitionen im Workflow auslösen. Es ist prinzipiell nur an Level mit Prozentsatz 100 % erlaubt, solche Attributänderungen zu spezifizieren

- Durch Unterelemente **<escalationMessage>** können die Empfängergruppe, die zu verwendende Nachrichtenvorlage und die in der Nachricht verlinkte Liste geändert werden.
  - Hat ein Level keine solchen Unterelemente, oder werden dort keine Empfängergruppen angegeben, so geht die Nachricht immer an die Task-Owner.
  - Hat ein Level keine solchen Unterelemente, oder werden dort keine Nachrichtenvorlagen angegeben, wird die Standardvorlage **monitorjob** verwendet. Eigene Nachrichtenvorlagen können einen beliebigen Text haben, müssen aber mindestens den Platzhalter **$monitorLog** enthalten.
  - Hat ein Level keine solchen Unterelemente, oder wird kein Link auf eine Liste angegeben, wird abhängig vom Objekttyp ein Standardlink geöffnet, der die Owner-Liste für diese Objekte aufruft.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Übernehmen Sie die entsprechenden Elemente **<escalation>** und fügen Sie neue Unterelemente **<level>** hinzu. Hat ein Level Unterelemente **<attributeChange>**, werden diese auf das Objekt angewendet. Hat ein Level Unterelemente **<escalationMessage>**, werden diese beim Verschicken der Nachrichten durch den Überwachungs-Job berücksichtigt.
Dokumente	escalations.xml, escalations.xsd
Beispiel	Escalations_ModifyLevels \WEB-INF\config\custom\xml\custom.xml

4.10.4 Updater

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Siehe Generator (Seite 81)
Dokumente	Commandchain_update.xml

4.11 Offline-Bearbeitung anpassen

Die Offline-Bearbeitung ist in ihrem Ablauf nicht anpassbar. Dies betrifft das Herunterladen in einem Formular und der Liste, die Zuordnung eines Offline-Bearbeiters, die Generierung eines oder mehrerer Offline-Dokumente, das Hochladen mit der manuellen oder automatischen Bestätigung usw. Bis zu einem gewissen Grad können die einzelnen Schritte aber inhaltlich verändert werden. Die folgenden Kapitel beschreiben die einzelnen Möglichkeiten als reine Konfiguration per XML oder Programmanpassung.

4.11.1 Offline-Dokumente ändern

Die Standardimplementierung der Offline-Bearbeitung nutzt die Report-Engine, um spezielle Excel-Dokumente während des Herunterladens zu generieren und während des Hochladens wieder einzulesen. Grundlage hierfür sind spezielle Reportdefinitionen, die in den Standard-Report-XMLs des Namens **reports_offlineprocessing_<Komponentenname>.xml** abgelegt sind. Diese Reportdefinitionen haben gegenüber den gewöhnlichen Reportdefinitionen zusätzliche Konventionen:

- Ihre ID wird immer in Großbuchstaben angegeben. Beispiel: **<ID des Objekttyps>_<ID der Bearbeiterrolle>**.
- Die Standardimplementierung nutzt ausschließlich Excel-Reporte, daher muss das Excel-Format verwendet werden.
- Die Excel-Zellen, die für den Offline-Bearbeiter bearbeitbar sein sollen, müssen mit dem Renderer **offlineProcessingInputReferenceRenderer** markiert werden; optional auch mit dem Style **offlineinputcell**.
- Die Autokomponente **offlineinfo** muss Teil der Report-Definition sein.

Bei der Anpassung der Reportdefinition hinsichtlich der editierbaren Zellen ist darauf zu achten, dass diese Zellen für die gewählte Bearbeiterrolle von der Rule-Engine auch tatsächlich als bearbeitbar markiert werden. Andernfalls wird jeder Versuch Objekte hochzuladen abgebrochen. Das folgende Beispiel zeigt, wie für den Tester die beiden Testfall-Attribute **Walkthrough name** und **Walkthrough counter** in die Offline-Bearbeitung aufgenommen werden können.

Speicherort	XML-Datei im Ordner **xml** DRL-Datei im Ordner **rules**.
Vorgehen	1. Kopieren Sie die Reportdefinition **TESTCASE_TESTER** des Formulars in eine eigene custom.xml-Datei und verwenden Sie für die Zellen der Attribute **Walkthrough name** und **Walkthrough counter** den Report-Renderer **offlineProcessingInputReferenceRenderer**. 2. Kopieren Sie die Regeln des Testfallformulars in eine eigene testcase.drl. 3. Passen Sie die Regeln so an, dass der Tester die beiden Attribute im Formular bearbeiten darf.
Dokumente	reports_offlineprocessing_testmanagement.xml als Vorlage. testcase.drl als Vorlage
Beispiel	▪ 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

4.11.2 Definition der Offline-Operator-Rollen ändern

In der Konfigurationsdatei **offlineProcessing.xml** ist festgelegt, welche Rollen der Offline-Bearbeitung als Operatoren von anderen Rollen auftreten können. Im Standard sind diese Operatoren für **Test-Management**, **Risiko-Management** und **Befragungs-Management** jeweils die Manager-Gruppen, die für ihre jeweiligen Owner- und Reviewer-Gruppen einen Checkout starten können. Um eine neue Rolle einzufügen, egal ob sie als Rolle für die Offline-Bearbeitung verwendet werden soll oder nicht, ist generell eine umfassende Anpassung von ARIS Risk & Compliance Manager und der Workflows notwendig. Daher zeigt das unten stehende Beispiel lediglich die Einschränkung der Operator-Rolle des Risiko-Managers, die ohne eine solche Neueinführung auskommt. Der Risiko-Manager ist in diesem Beispiel nur noch Operator für den Risiko-Owner, nicht mehr für den Risiko-Reviewer.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Kopieren Sie das XML-Element **<offline-operators>** aus der Datei **offlineProcessing.xml**. Behalten Sie nur den gewünschten <operator-role>-Eintrag bei und ändern ihn.
Dokumente	offlineProcessing.xml als Vorlage.
Beispiel	OfflineProcessing_ChangeOperator\WEB-INF\config\custom\xml\custom.xml: Restrict operator role of risk manager

4.11.3 Neue Offline-Bearbeiter-Rolle hinzufügen

In der Konfigurationsdatei **offlineProcessing.xml** ist festgelegt, welche existierenden Rollen in ARIS Risk & Compliance Manager als Rollen für die Offline-Bearbeitung für bestimmte Objekte fungieren können. Im Standard sind diese Rollen für **Test-Management**, **Risiko-Management** und **Befragungs-Management** jeweils die Manager-, Owner- und Reviewer-Gruppen. Um eine neue Rolle einzufügen, ist generell eine umfassende Anpassung von ARIS Risk & Compliance Manager und der Workflows notwendig. Daher zeigt das unten stehende XML-Fragment (Auszug aus einer custom.xml) lediglich das Hinzufügen einer neuen Rolle **testvalidator** ohne jede weitere Einbindung in die bestehenden 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>
```

4.11.4 Anpassung der Offline-Prozessoren

Im Beispiel **Offline-Dokumente ändern** (Seite 86) wird erläutert, wie die Excel-Dokumente der Offline-Bearbeitung, die durch die Report-Engine generiert werden, angepasst werden können. Es ist auch möglich, die Excel-Dateien auf komplett andere Weise zu generieren oder auch Dokumente anderer Formate zu verwenden, indem die Standard-Offline-Prozessoren durch eigene Implementierungen ersetzt werden, die jeweils solche Dokumente generieren und die Änderungen des Offline-Bearbeiters darin wieder auslesen und in ARIS Risk & Compliance Manager zurückschreiben. Diese Klassen müssen das Interface **ICheckOutProcessor** bzw. **ICheckInProcessor** implementieren und können dann in einer **custom.xml** durch das unten stehende XML-Fragment eingebunden werden:

```
<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>
```

4.11.5 Anpassung des Offline-Verhaltens pro Objekttyp

Pro Objekttyp wird die Entscheidung, wann ein bestimmtes Objekt abhängig von seinem Zustand und dem aktuellen Benutzer als offline bearbeitbar eingestuft wird, von der ihm zugeordneten Implementierung des Interfaces **IOfflineProcessingBehaviour** getroffen. Im Standard besitzt beispielsweise der Fragebogen die Besonderheit, dass alle untergeordneten Kapitel, Fragen usw. ebenfalls zur Bearbeitung gesperrt werden, sobald für den Fragebogen die Offline-Bearbeitung gestartet wird. Dieses Verhalten kann angepasst werden, indem eigene Implementierungen des oben genannten Interfaces in einer custom.xml durch das unten stehende XML-Fragment eingebunden werden.

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

4.12 Dashboard-Link hinzufügen / anpassen

4.12.1 DashBoard-Link anpassen

Wird für das Customizing ein Objekttyp angepasst, z. B. ein Attribut ergänzt, soll diese Änderung in der Regel auch in der bestehenden MashZone-Liste nachgezogen werden. Hierzu muss die entsprechende MashZone-Datenabfrage in den Customizing-Bereich kopiert und entsprechend modifiziert werden.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	1. Fügen Sie unter dem Sammelelement **<custom>** das bestehende Element **<view>** als Kopie aus der Standardkonfiguration ein. 2. Führen Sie die gewünschten Änderungen durch, z. B. zusätzliche Attribute einfügen oder unerwünschte Attribute entfernen.
Dokumente	mashzone_views.xml, views.xsd
Beispiel	▪ 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

4.12.2 Dashboard-Link hinzufügen

4.12.2.1 MashZone-Liste für Objektdaten hinzufügen

Um in ARIS Risk & Compliance Manager in **Administration > Importe und Exporte > Dashboard-Link generieren** eine neue Liste hinzuzufügen, muss in der Konfiguration eine neue MashZone-relevante Datenabfrage (**view**) angelegt werden. Die Konfiguration von Datenabfragen wird detailliert im zugehörigen XML-Schema **views.xsd** beschrieben. Die Listen- und Filteransicht wird im Link-Generator automatisch erzeugt, wenn beim entsprechenden Element **<view>** das Attribut **relevantForMashzoneIntegration** den Wert **true** hat. Dann ist auch über die MashZone-Schnittstelle eine CSV-Abfrage möglich.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Legen Sie unter dem Sammelelement **<custom>** ein neues Element **<view>** an. Für MashZone-Datenabfragen gelten die gleichen Regeln wie für die übrigen Datenabfragen. Sie müssen jedoch durch das Flag **relevantForMashzoneIntegration="true"** gekennzeichnet werden. Es können so z. B. auch bestehende Datenabfragen durch Kopieren und Setzen des true-Flags schnell in MashZone-Datenabfragen umgewandelt werden. Im URL-Generator sind später alle Spalten (**<viewsColumn>**) verfügbar, die nicht durch das Setzen des Flags **mashzoneRelevant="false"** explizit ausgeschlossen werden.
Dokumente	mashzone_views.xml, views.xsd
Beispiel	▪ 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

4.12.2.2 MashZone-Liste für Objektverknüpfungen hinzufügen

Das Vorgehen beim Hinzufügen einer Liste für Objektverknüpfungen entspricht in allen Details dem Hinzufügen einer Datenliste. Der Unterschied liegt in den Inhalten. Stehen bei der Datenliste die Datenattribute des Objekts im Vordergrund, sind es hier die IDs der zu verknüpfenden Objekte. Es wird empfohlen, sich an einer bestehenden Liste zu orientieren.

Speicherort	XML-Datei im Ordner **xml**
Vorgehen	Generieren Sie eine View, die nur die IDs und gegebenenfalls den Namen der zu verknüpfenden Objekte ausgibt. Die Objekte können dann innerhalb des MashZone-Feed-Editors verknüpft werden.
Dokumente	mashzone_views.xml, views.xsd
Beispiel	▪ 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

4.12.2.3 MashZone-Liste benennen

MashZone-Listen können über einen Property-Schlüssel in einer Property-Datei mit einem Namen versehen werden. Es gilt dabei folgende Namenskonvention:

- Der Schlüssel muss das Format **view.<view ID>.DBI** haben.
- Die Datei darf keinen Unterstrich (_) als Trennzeichen enthalten, da der Unterstrich als Trennzeichen für die Länderkennung reserviert ist.

Soll der Name in verschiedenen Sprachen verfügbar sein, muss pro Sprache eine Datei generiert und mit dem Länderkennzeichen versehen werden. Beispiel: **custom_de.properties** (Deutsch) oder **custom_en.properties** (Englisch).

Speicherort	Property-Datei im Ordner **properties\application**
Vorgehen	Fügen Sie in einer neuen oder bestehenden Datei entsprechend der o. g. Namenskonvention eine Zeile hinzu und tragen Sie hinter dem Gleichheitszeichen den gewünschten Namen ein.
Dokumente	Siehe **Namen anpassen** (Seite 4).
Beispiel	AddModifyMashzoneURL \WEB-INF\config\custom\properties\application\custom.properties: Add new data list und Add new object relation list

4.13 Navigation anpassen

Die Navigation innerhalb der HTML-Oberfläche von ARIS Risk & Compliance Manager kann durch Anpassen der Navigation-XML-Dateien teilweise angepasst werden. Für diese Navigationsbereiche sind Anpassungen möglich:

- Die Navigation innerhalb der einzelnen Bereiche (derzeit nur **Home**, **Explorer** und **Auswertungen**)

4.13.1 Navigation für einen Bereich anpassen

Der Inhalt von **Home, Explorer und Auswertung** kann über die zugehörige XML-Datei angepasst werden (navigation_home.xml, navigation_explorer, navigation_evaluation). Für **Home** ist in der XML-Datei festgelegt, welche Menüpunkte im Hauptfenster angezeigt werden. Für **Explorer** und **Auswertung** ist in der XML-Datei festgelegt, wie die linke Navigation aufgebaut ist.

Innerhalb der Navigation-XML-Dateien können folgende Elemente definiert werden:

- **<nav.item>**

  Wird als ein Gliederungselement für eine Gruppe von Elementen verwendet. Alternativ kann es als eine Referenz für ein Element verwendet werden, das in den XML-Dateien definiert ist.
- **<nav.data.grid>**

  Wird als Link dargestellt und öffnet eine Liste. Wird in allen Bereichen verwendet
- **<nav.evaluation>**

  Wird als Link dargestellt und öffnet eine Auswertung. Wird nur in **Auswertung** verwendet

Die Anzeige dieser Elemente kann per <nav.access> definiert werden. Das Element wird dann nur angezeigt, wenn alle im <nav.access> definierten Bedingungen erfüllt sind. Es gibt folgende Bedingungsarten:

- **<nav.access.component>**

  Der Benutzer muss in ARIS Risk & Compliance Manager der Rolle zugeordnet sein, die ihm den Zugriff auf die für ihn relevanten Funktionen gibt.
- **<nav.access.privilege>**

  Der Benutzer muss mit mindestens einer ihm zugeordneten Rollen das angegebene Systemrecht besitzen. Die Rechte sind in der Datei **roles.xml** definiert.
- **<nav.access.role>**

  Der Benutzer die angegebene Rolle besitzen. Die Rechte sind in der Datei **roles.xml** definiert.

In jeder Bedingung kann jede Bedingungsart nur genau einmal vorkommen.

Ein Element **<nav.item>** vererbt seine Zugangsbedingungen an alle untergeordneten Elemente. Beinhaltet ein Element **<nav.item>** ein untergeordnetes Element **<nav.data.grid>**, werden die Bedingungen aus den direkt zugeordneten und denen des übergeordneten Elements kombiniert.

Beispiel

In der Navigation wird eine bestehende Liste an einer weiteren Stelle hinzugefügt und es wird eine andere Zugangsberechtigung dafür definiert.

Speicherort	XML-Datei im Ordner xml
Vorgehen	Kopieren Sie das Element **<nav.module>** oder einzelne Elemente **<nav.item>** aus dem Standard in das Customizing. Und umschließen Sie sie mit einem <navigation> Element. Danach kann der Inhalt mithilfe der Datei **navigation.xsd** geändert werden.
Dokumente	navigation_explorer.xml, navigation.xsd
Beispiel	Navigation_Module\WEB-INF\config\custom\xml: Modify the module navigation

4.14 Event-Enabling anpassen und erweitern

Die Steuerung von Abläufen durch Events erfolgt durch die Verarbeitung von eingehenden Events und löst die Generierung von bestimmten Objekten in verschiedenen Status in ARIS Risk & Compliance Manager aus. Diese Steuerung kann nicht angepasst werden. Angepasst werden können die Attribute der generierten Objekte von ARIS Risk & Compliance Manager, z. B. Testfälle, indem die vorhandene Konfigurationsdateien angepasst werden oder indem eine zusätzliche Konfigurationsdatei auf Grundlage der vorhandenen generiert wird. Diese Anpassungen und Erweiterungen werden durch einen Administrator der Complex Event Processing Engine durchgeführt. Im Folgenden werden zwei Anpassungsmöglichkeiten in ARIS Risk & Compliance Manager beschrieben. Informationen darüber, wie die Dateien in der Complex Event Processing Engine verwaltet werden, d. h. welche Events über welchen Weg und auf Basis welcher Konfigurationsdatei gesendet werden, entnehmen Sie bitte der Dokumentation von Complex Event Processing.

4.14.1 Bestehende Event-Type-XSDs erweitern

Die Standard-Implementierung der Steuerung durch Events verwendet mitgelieferte Konfigurationsdateien in Form von XSDs für die Complex Event Processing Engine. Diese Dateien enthalten alle Attribute, die das jeweilige Event in ARIS Risk & Compliance Manager übertragen kann. Sobald der Administrator der Complex Event Processing Engine die Änderungen in der Konfigurationsdatei speichert, enthalten die Events die neuen Informationen. Ggf. muss die Konfigurationsdatei lokal aktualisiert werden.

Speicherort	XSD-Datei im Ordner **<event architecture installation directory>\common\EventTypeStore\WebM\ARCM**
Vorgehen	Fügen Sie der Konfigurationsdatei ein entsprechendes neues Attribut hinzu. Sollte es sich um ein Attribut aus ARIS Risk & Compliance Manager handeln, muss der gleiche Name wie in ARIS Risk & Compliance Manager verwendet werden.
Dokumente	objectTypes.xml, IncidentEvent.xsd oder TestcaseEvent.xsd

4.14.2 Neue Event-Type-XSDs anlegen

Die Standard-Implementierung der Steuerung durch Events verwendet mitgelieferte Konfigurationsdateien in Form von XSDs für die Complex Event Processing Engine. Diese Dateien enthalten alle Attribute, die das jeweilige Event in ARIS Risk & Compliance Manager übertragen kann. Zusätzlich besteht die Möglichkeit neue Konfigurationsdateien auf Basis einer der mitgelieferten Konfigurationsdateien zu generieren. Der Objekttyp der in ARIS Risk & Compliance Manager generiert werden soll kann in der Konfigurationsdatei nicht geändert werden. Der Administrator muss dann nicht nur die neue Konfigurationsdatei speichern, sondern auch dafür sorgen, dass über einen entsprechenden Weg Events auf Basis der neuen Konfigurationsdatei verschickt werden. Ggf. muss die neue Konfigurationsdatei im lokal verfügbaren Event Type Store aktualisiert werden.

Speicherort	XSD-Datei im Ordner **<event architecture installation directory>\common\EventTypeStore\WebM\ARCM**
Vorgehen	Kopieren Sie die vorhandene XSD-Datei im Ordner **ARCM** innerhalb des jeweiligen Event Type Store und ergänzen Sie sie durch weitere Attribute. Sollte es sich um ein Attribut aus ARIS Risk & Compliance Manager handeln, muss der gleiche Name wie in ARIS Risk & Compliance Manager verwendet werden.
Dokumente	objectTypes.xml, IncidentEvent.xsd oder TestcaseEvent.xsd als Vorlage