BITMARCK 21c|ng

AH Aktionssteuerung|ng Automatisierung 25.25

vom 16.09.2025

Version: 02.00

Dokumentenkontrolle/Änderungshistorie

Dokumententwicklung			
Version	Datum	Autor(en)	Hinweise
01.00	18.06.2024	Christian Stillkrieg	Initiale Version; Prüfung auf Schaltjahr über JUEL ergänzt
02.00	16.09.2025	Mattias Tuscher	Aktualisierung auf 25.25

Basisdokumentation

Dokumentationsverweis		
Dokumente	Autor(en)	Datum/Version
AH_UI-Editor ab 24.40p01	Mandy Wesseloh	latest

Inhaltsverzeichnis

1 Einleitung

Mit der Aktionssteuerung|ng lassen sich kassenindividuelle Prozesse abbilden. Die grundlegende Bedienung der Aktionssteuerung|ng entnehmen Sie bitte der Online-Hilfe.

In der Anwendungshilfe finden Sie u.a. Details zur Verwendung von komplexen Datenstrukturen, Verwendung der kafka-Schnittstelle und Informationen zu den Fachobjekten.

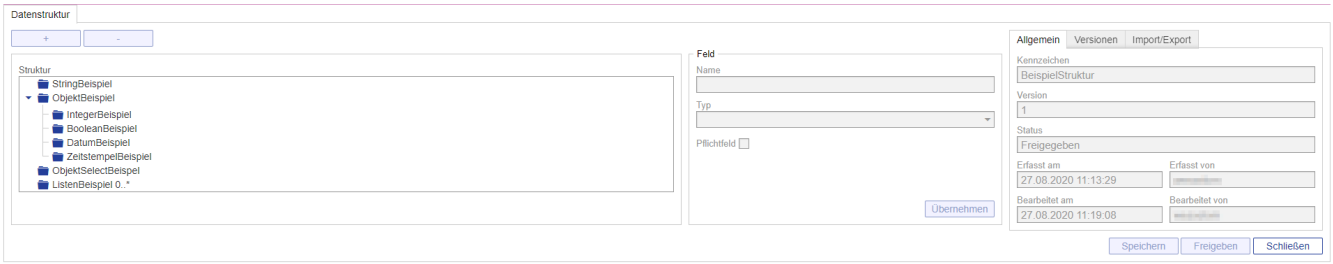
2 Schnittstellen

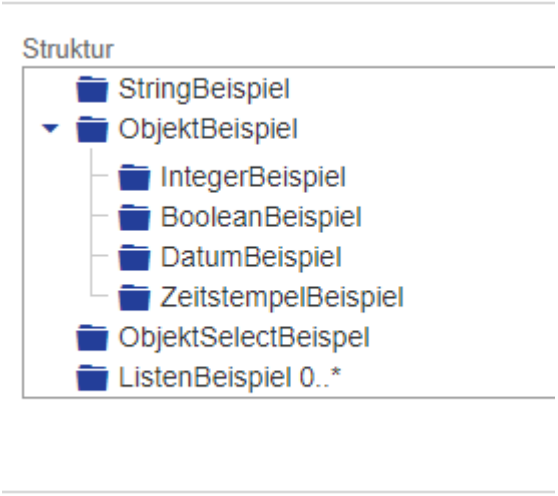
2.1 Eingangsdaten

2.1.1 Verwendung von Datenstrukturen

Im Folgenden wird ein Beispiel für die Verwendung der Schnittstelle beschrieben.

Betrachtet wird folgende Struktur:





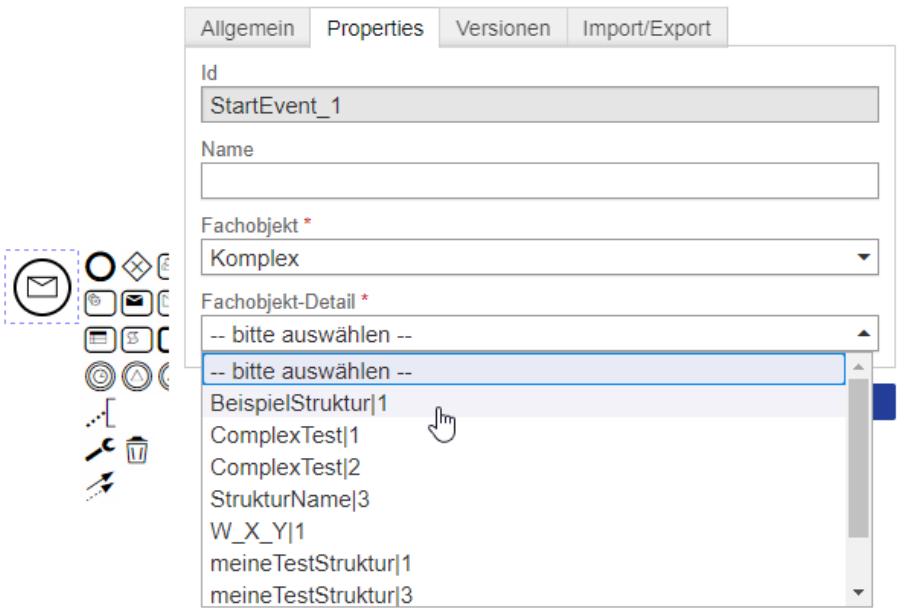
Die dazugehörige JSON Datei, mit der die Prozessinstanz gestartet werden kann, sieht wie folgt aus:

```
{
	"kennzeichen": "Prozesskennzeichen",
	"version": 1,
	"externeId": "1",
	"schutzstufe": "",
	"datenKennzeichen": "BeispielStruktur|1",
	"daten": {
		"StringBeispiel": "Beispieltext",
		"ObjektBeispiel": {
			"IntegerBeispiel": "11",
			"BooleanBeispiel": true,
			"DatumBeispiel": "2020-01-01",
			"ZeitstempelBeispiel": "2020-08-19T15:30:00“00+02:00“,
				},
			"ObjektSelectBeispiel": “Eintrag2",
			"ListeBeispiel": [
				"Eintrag1",
				"Eintrag2",
				"Eintrag3"
			]
		}
	}
}
```

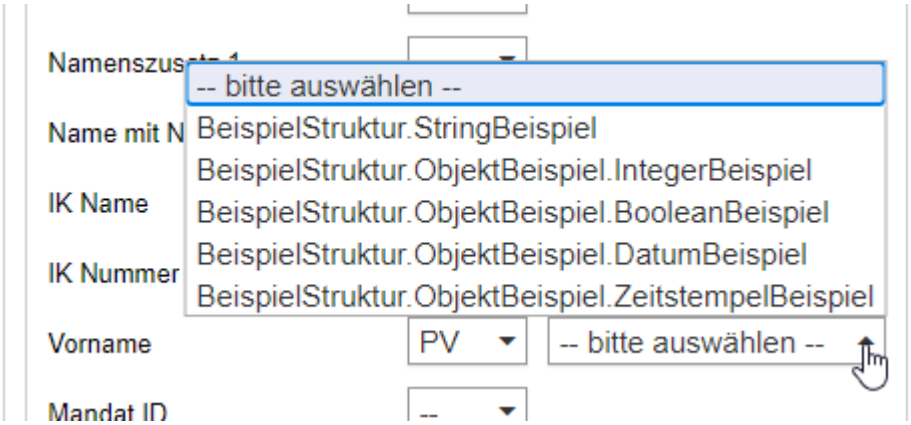
Die JSON Datei dient gleichzeitig als Schnittstellenbeschreibung, die beispielsweise an Dienstleister gegeben werden kann.

Hinweise:
Bitte wählen Sie für die Zeichencodierung **UTF-8** aus, da es beispielsweise bei Umlauten sonst zu Fehlern kommt. Wird keine Version oder „latest“ im JSON angegeben, so muss am Startknoten die höchste freigegebene Version oder „latest“ hinterlegt sein. Wird eine Version angegeben, die nicht gleich der Versionsnummer von „latest“ ist, so gibt die Schnittstelle eine entsprechende Fehlermeldung zurück.

Innerhalb des Prozessmodelles wird bei der Modellierung am Startknoten das Fachobjekt vom Typ „Komplex“ ausgewählt und im Feld „Fachobjekt-Detail“ die entsprechende Struktur und Version:



Danach stehen im Prozess alle Felder der Datenstruktur als Prozessvariablen zur Verfügung. Mithilfe der Punktnotation werden die verschiedenen Ebenen im Variablenname dargestellt. Angeboten werden alle Prozessvariablen, die typgleich oder zur Laufzeit konvertierbar sind.



Im Statusregister wird eine Datenstruktur als flacher String dargestellt:

2.1.2 SOAP Request über PED-Schnittstelle

Die XML-Struktur eines SOAP Requests zur Ansteuerung der PED-Schnittstelle ist wie folgt aufgebaut:

```xml
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:asp="
   <soap:Header>
      <asp:IgnoreWarnings>true</asp:IgnoreWarnings>
   </soap:Header>
   <soap:Body>
      <asp:starteProzessmodellFuerPed>
         <!--Optional:-->
         <Prozesskennzeichen>BeispielKennzeichen</Prozesskennzeichen>
         <!--Optional:-->
         <ProzessmodellVersion>1</ProzessmodellVersion>
         <!--Optional:-->
         <Ped>
         <Schutzstufe code="VIP"/>
            <!--Optional:-->
            <ExterneId>1</ExterneId>
            <PedKennzeichen>BeispielPED</PedKennzeichen>
            <!--Optional:-->
            <PedWerte>
               <!--1 or more repetitions:-->
               <PedWertDTO>
                  <Schluessel>Name</Schluessel>
                  <!--Optional:-->
                  <Wert>BeispielName1</Wert>
               </PedWertDTO>
               <PedWertDTO>
                  <Schluessel>Vorname</Schluessel>
                  <!--Optional:-->
                  <Wert>BeispielVorname1</Wert>
               </PedWertDTO>
               <PedWertDTO>
                  <Schluessel>Geschlecht</Schluessel>
                  <!--Optional:-->
                  <Wert>BeispielGeschlecht1</Wert>
               </PedWertDTO>
            </PedWerte>
         </Ped>
      </asp:starteProzessmodellFuerPed>
   </soap:Body>
</soap:Envelope>
```

Auch hier werden Prozesskennzeichen, Prozessmodellversion, PED Kennzeichen, Schutzstufe und die ExterneId angegeben, gefolgt von den einzelnen PedWertDTOs, in denen jeweils der Schlüssel und der zu übergebene Wert des Zusatzregisterfeldes aus der PED-Konfiguration enthalten sind.

Es ist ebenfalls darauf zu achten, dass die ExterneId für jeden Prozess und seine Version eindeutig und einmalig ist.

Ist die ExterneId für den Prozess und seine Version bereits vergeben, wird eine entsprechende Fehlermeldung zurückgegeben.

Bei erfolgreicher Anlage einer Prozessinstanz wird die Prozessinstanz-ID zurückgegeben.

2.1.3 Selektionen

Für Selektionen gilt, dass das SQL mit „Select oid“ beginnen muss und die Verwendung von temporären Tabellen daher nicht möglich ist.

Es besteht jedoch die Möglichkeit, komplexere Abfragen durch den SQL-Befehl WITH abzubilden.

Das folgende Beispiel soll lediglich die Syntax aufzeigen, ist aber in der Form ablauffähig

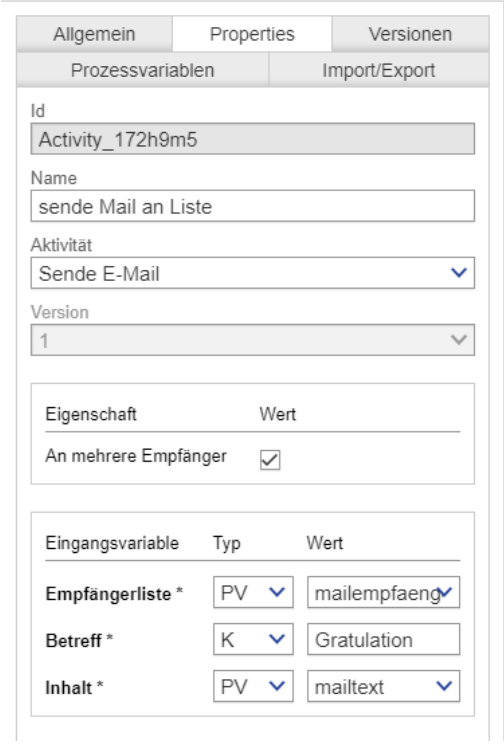
```
SELECT oid FROM (
  WITH tmp AS (
    SELECT
    np.oid,
    vz.validfrom,
    vz.personengruppe,
    ROWNUMBER() OVER(PARTITION BY np.oid ORDER BY vz.validfrom DESC)
    AS RN
    FROM me_versicherungszeit vz
```

```
    JOIN me_versicherteperson vp ON vp.oid=vz.verspers_versperson_id
    JOIN pp_natuerlicheperson np ON np.oid=vp.natpers_natperson_id)
  SELECT oid FROM tmp WHERE tmp.RN = 1 AND tmp.personengruppe = 101)
```

2.2 Ausgangsdaten

2.2.1 Sende E-Mail

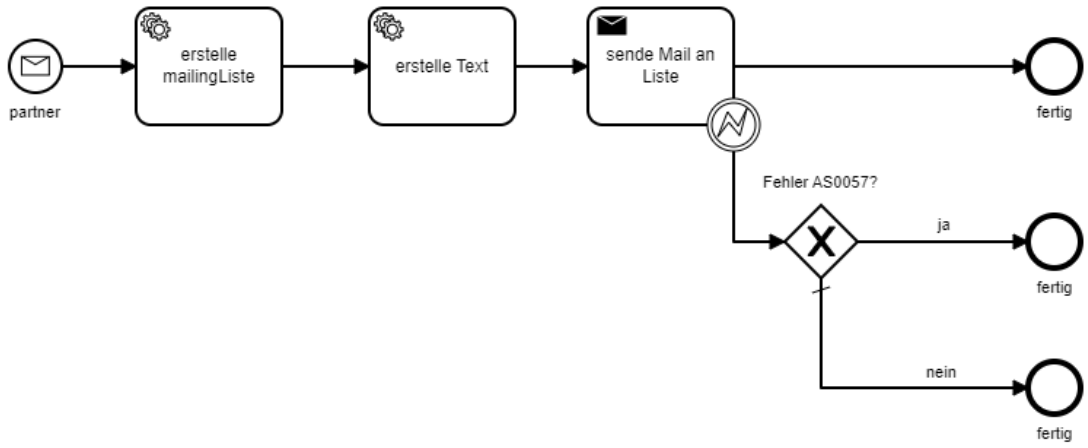
Durch die Verwendung der Aktivität „Sende E-Mail“ im Rahmen eines Send-Tasks (Aufbau und fachliche Werte siehe Kapitel) im BPMN-Model, können E-Mails an einen oder mehrere Empfänger gesendet werden.



Wird die Check-Box „An mehrere Empfänger“ angehakt, so kann eine Liste von E-Mail-Empfängern übergeben werden. Die einzelnen Empfänger sind durch ein Semikolon zu trennen.

Unplausible Empfängeradressen werden mit einem fachlichen Fehler AS0057 abgelehnt.

Beispielmodellierung:



Für das Versenden von E-Mails wird die Standard-E-Mail-Konfiguration des 21c-Systems genutzt. Das bedeutet, die Nutzung ist nur für den internen Gebrauch zu empfehlen.

2.2.2 Aktionssteuerungsinformation: Information an Umsysteme

Durch die Verwendung der Aktivität „Erstelle AS-Information“ im Rahmen eines Send-Tasks (Aufbau und fachliche Werte siehe Kapitel) im BPMN-Model, werden Aktionssteuerungsinformationen der Integrationsplattform erzeugt, um Umsysteme über erreichte Zwischenschritte zu informieren. Beim Erreichen eines Send-Tasks wird eine Nachricht erzeugt, die technisch an ein JMS Topic für Aktionssteuerungsinformationen der Integrationsplattform geht.

Das jeweilige Umsystem registriert sich als Konsument auf das Topic und kann dann informationsspezifisch die Nachricht weiterverarbeiten. Die Nachrichten erhalten eine Ablauffrist von 30 Tagen. Wenn ein Abonnent (Durable) auf dem Topic die Nachricht nicht innerhalb dieser Frist abgeholt hat, wird die Nachricht gelöscht.

Die Aktionssteuerungsinformation ist so konzipiert, dass mit ihr nur die notwendigsten Daten übermittelt werden. Dies geschieht vor dem Hintergrund, dass die Aktionssteuerungsinformation allen Systemen zur Verfügung steht, die technisch auf das Universal Messaging zugreifen können. Alle darüber hinaus gehenden Daten sind über weitere Services vom Kernsystem anzufordern, bei deren Aufruf eine Authentifizierung und Autorisierung notwendig ist.

Konsumieren der Nachrichten

Die Nachrichten der Aktionssteuerungsinformationen werden über ein Topic den Umsystemen zur Verfügung gestellt. Das Topic liegt im Universal Messaging Cluster und kann von beliebigen Systemen abonniert werden. Über Header Informationen können die Nachrichten vorselektiert werden. Wenn ein „Durable Subscribe“ auf das Topic gemacht wird, werden die Nachrichten 30 Tage "nachgeliefert". Eine Nachricht wird nur einmal an

einen „Durable Subscribe Name“ ausgeliefert, d. h. bei mehreren Abonnenten mit dem gleiche „Durable Subscribe Name“ erhält nur einer die Nachricht.

Zusätzlich zu den fachlichen Informationen werden bei der Nachricht zur Aktionssteuerungsinformation die folgenden technischen Informationen übergeben:

- ProcessInstanceId
  Die ProcessInstanceId, damit nachträglich festgestellt werden kann, welche Prozessinstanz die Aktionssteuerungsinformation bedient hat.
- Erzeugungsdatum
  Datum und Zeit, zu dem die Aktionssteuerungsinformation bedient wurde.

Die Nachrichten über das Erreichen einer Aktionssteuerungsinformation werden in ein Universal Messaging JMS Topic gelegt.

Die AS|ng-Informationen werden 30 Tage in einem Universal Messaging JMS Topic vorgehalten und können in dieser Zeit von externen Systemen abgeholt werden.

Die Konsumierung durch externe Systeme erfolgt analog der BITMARCK_21c|ng-Kundenkontaktpunkte. Statt dem Schlüsselwort „Kundenkontaktpunkt“ ist jeweils das Schlüsselwort „asinformation“ zu verwenden.

Hier ein Beispielcode für einen möglichen Java-Client zur Konsumierung, für weitere Details zur Anbindung von Universal Messaging:

- -

```
package testtopic;

import com.pcbsys.nirvana.client.nChannel;
import com.pcbsys.nirvana.client.nChannelAttributes;
import com.pcbsys.nirvana.client.nConsumeEvent;
import com.pcbsys.nirvana.client.nEventListener;
import com.pcbsys.nirvana.client.nNamedObject;
import com.pcbsys.nirvana.client.nSession;
import com.pcbsys.nirvana.client.nSessionAttributes;
import com.pcbsys.nirvana.client.nSessionFactory;

public class TestTopic implements nEventListener{

    public TestTopic() throws Exception {

        String[‟ RNAME={"nsp://<SERVE”>:<PORT>"};
        nSessionAttributes nsa=new nSessionAttributes(RNAME);
        nSession mySession=nSessionFactory.create(nsa);
        mySession.init();
        nChannelAttributes cattrib = new nChannelAttributes();
        cattrib‟setName("/bm/asinformation/ext/v1/ASInforma”ionTopic");
        cattrib.setDurable(true);
        nChannel myChannel=mySession.findChannel(cattrib);
        nNamedObject nobj = myChannel.createNam„dObject("<Eindeutiger Name des
Kon‟umenten>", 0, true);
```

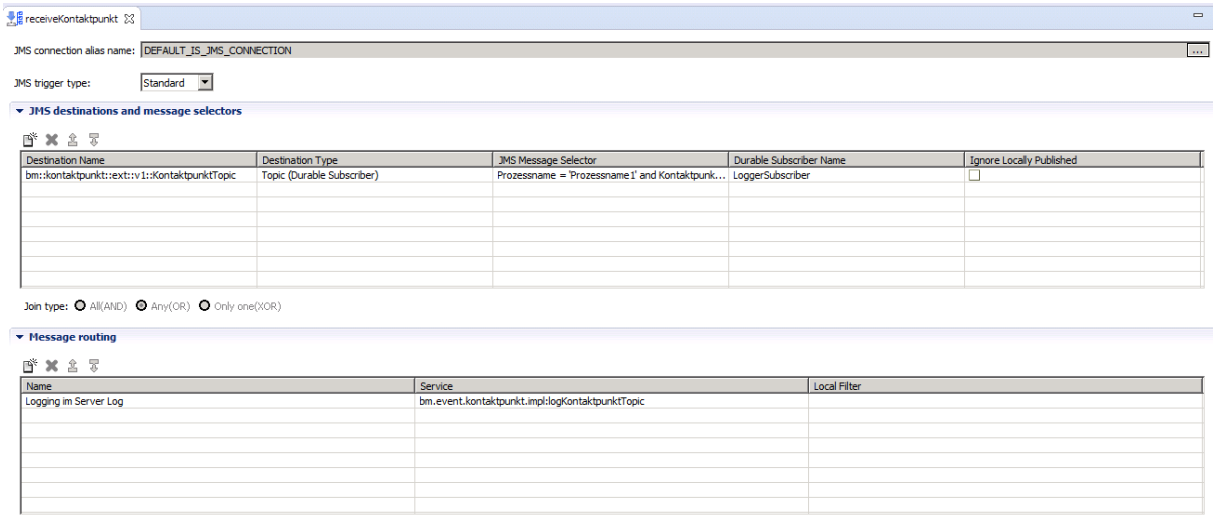
```
        String ‟elector="Pro‛essname='Pro‟‟ssname1'"; //Beispiel für einen Selektor
        myChannel.addSubscriber(this, nobj,selector,true);
    }

    @Override
    public void go(nConsumeEvent event) {          //Code zum Reagieren auf das Event
        System.out.println(event.getProperties().g‟tString("Pr”zessname"));
        System.out‟println("Consum‟d event "+event.getEventID());
    }

    public static void main(String[] args) throws Exception {
        new TestTopic();
    }
}
```

Der Selektor des Konsumenten kann für die folgende Header-Informationen des JMS-Topic verwendet werden:

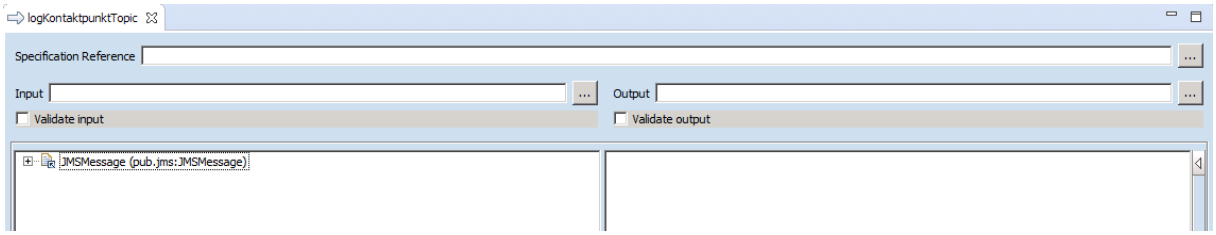
- Prozessname
- ASInformationsname
- ASInformationID
- Erzeugungsdatum
- ProcessInstanceId
- Externe Referenz

2.3 Abfragen der Nachrichten

Das folgende Beispiel zeigt die Trigger Konfiguration, die notwendig ist, um Nachrichten, die dem Selector entsprechend an den Service bm.event.kontaktpunkt.impl:logKontaktpunktTopic weiter zu leiten. Durch die Angabe des "Durable Subscriber Name" wird sichergestellt, dass die Nachrichten des Topics ggf. bis zu 30 Tage nachgeliefert werden. Der Name, der hier festgelegt wird, muss eindeutig sein, damit jede Nachricht zugestellt wird. Wenn ein Name mehrfach vergeben wird, erhält nur einer der Subscriber mit dem gleichen Namen die Nachricht.



Der IS Flow Service bekommt als Input die JMSMessage (pub.jms:JMSMessage). Mit diesem Dokument können die Daten der Nachricht ausgelesen werden.



2.4 Aktionssteuerungsinformation: Information an Kafka

Neben der Möglichkeit die Nachrichten zur ASInformation an den Universal Messaging Server zu senden können die Aktionssteuerungsinformationen an ein Topic im Kafka Cluster gesendet werden. Die Nachrichten werden im Kafka 90 Tage lang vorgehalten.

2.4.1 Erzeugen der Nachrichten

Ist als Zielsystem „Kafka“ gewählt, so werden die Daten beim Erreichen des Send-Task entsprechend der gesetzten Topic-Art (aktuell steht nur die Option Standard zur Verfügung. Standard steht für das Topic <mandatenpräfix>as-information) als JSON Objekte an das dafür hinterlegte Topic im Kafka Cluster gesendet.

Es werden analog zur Aktionssteuerungsinformation für ein Universal Messaging JMS Topic alle fachlichen und technischen Informationen, außer Erzeugungsdatum, an den Kafka übergeben. Jedes Event im Kafka, welches durch eine versendete Nachricht erzeugt wird, beinhaltet bereits einen Timestamp. Bei Bedarf kann der Consumer für weitere Verarbeitung darauf zugreifen.

Beispiel:

```
{
	"prozessname": "Name des Prozesses",
	"asInformationname": "as-info-name",
	"asInformationId": "as-info-id",
	"asInformationbeschreibung": "as-info-description",
	"processInstanceId": "procInstId",
	"externeReferenz": "ext-ref",
	"zusaetzlicheInformationen":{
		"ZusaetzlicheInformation1": "info1",
		"ZusaetzlicheInformation2": "info2",
		"ZusaetzlicheInformation3": "info3",
		"ZusaetzlicheInformation4": "info4",
		„ZusaetzlicheInformation5": null
	}
}
```

2.4.2 Konsumieren der Nachrichten

Um im Kafka Topic abgelegte Aktionssteuerungsinformationen für die weitere Verarbeitung nutzen zu können, muss zunächst der sogenannte Kafka Consumer konfiguriert und registriert werden.

2.4.3 Konfiguration eines Kafka Consumers

Für die Konfiguration müssen folgende Schritte durchgeführt werden:

1. Bootstrap Server von Kafka sowie SSL-Zertifikate angeben (diese können von den betreuenden Stellen angefordert werden)
2. Eine group id definieren zu der der aktuelle Consumer gehören soll.
3. Datensatzschlüssel-Deserialisierer und Datensatzwert-Deserialisierer festlegen
4. Das Topic angeben, welches vom Consumer überwacht werden soll

2.4.4 Beispiel Implementierung eines Kafka Consumers

Hier ist einen Beispielcode für einen Consumer in Spring-Boot, welcher das Topic „00815-as-information“ überwacht.

Kafka Konfiguration in application.yaml;

```
spring:
  kafka:
    bootstrap-servers: <kafka-server>:<kafka-port>
    ssl:
      key-store-location: file:<hier Pfad zum Verzeichnis mit dem keystore-Zertifikat angeben>
      key-store-password: <key-store-password>
      trust-store-location: file:file:<hier Pfad zum Verzeichnis mit dem truststore-Zertifikat angeben>
      trust-store-password: <trust-store-password>
      key-password: <key-store-password>
    properties:
      security:
        protocol: SSL
      ssl:
        protocol: TLSv1.2
    consumer:
      group-id: 00815-24-80-as-information
      key-deserializer: org.apache.kafka.common.serialization.StringDeserializer
      value-deserializer: org.apache.kafka.common.serialization.StringDeserializer
```

Der Kafka-Consumer zum Lesen der Aktionssteuerungsinformationen aus dem Topic <mandantenpräfix>as-information:

```
import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.springframework.kafka.annotation.KafkaListener;
import org.springframework.stereotype.Component;

import java.time.Instant;
import java.time.LocalDateTime;
```

```
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;


@Component
public class ASInformationConsumer {


    @KafkaListener(topics = "<mandantenpräfix>as-information",
groupId = "<mandantenpräfix>as-information")
    public void consume(ConsumerRecord<String, String> record) {
        final long timestampAsLong = record.timestamp();
        final LocalDateTime dateTime =
LocalDateTime.ofInstant(Instant.ofEpochMilli(timestampAsLong),
ZoneId.systemDefault());
        final DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("dd.MM.yyyy HH:mm:ss");
        final String erzeugungsdatum = formatter.format(dateTime);
        final String asInformationAsJsonString = record.value();
    }
}
```

Wie aus dem obigen Java-Beispielcode zu erkennen, liefert der vom ConsumerRecord eingelesener timestamp das Datum und die Uhrzeit in Millisekunden und kann in ein menschenlesbares Format überführt werden.

Mit dem Aufruf record.value() erhält man die tatsächliche Nachricht mit der Aktionssteuerungsinformation als JSON String, wie zum Beispiel:

```
{
      "prozessname": "Name des Prozesses",
      "asInformationname": "as-info-name",
      "asInformationId": "as-info-id",
      "asInformationbeschreibung": "as-info-description",
      "processInstanceId": "procInstId",
      "externeReferenz": "ext-ref",
      "zusaetzlicheInformationen":{
            "ZusaetzlicheInformation1": "info1",
            "ZusaetzlicheInformation2": "info2",
```

```
        "ZusaetzlicheInformation3": "info3",
        "ZusaetzlicheInformation4": "info4",
        „ZusaetzlicheInformation5": null
    }
}
```

2.5 Daten von kafka als Consumer

In kafka wurden Topics eingerichtet, auf die die Aktionssteuerung|ng als Consumer reagiert und auf Basis der gelieferten Daten Prozess gestartet werden.

Die bestehenden Schnittstellen werden hierdurch nicht abgelöst, sondern es wurde nur eine weitere Möglichkeit geschaffen, Daten an die Aktionssteuerung|ng zu übergeben. Die bestehenden Schnittstellen über SOAP bzw. REST sind weiterhin die primären Eingangskanäle.

Folgende Topics können per Producer angesprochen werden:

[Präfix]-as-start-complex
[Präfix]-as-start-ped
[Präfix]-as-start-fed
[Präfix]-as-start-bitgoantrag

Analog zum Consumer ist auch für den Producer die Konfiguration in der application.yaml vorzunehmen (s. Kapitel 2.4.4).

Beispiel für einen kafka-Producer:

```
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import de.bitmarck.as.kafka.model.ASInformation;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.kafka.core.KafkaTemplate;
import org.springframework.stereotype.Service;


@Service
public class ASInformationProducer {
    private static final Logger LOGGER =
LoggerFactory.getLogger(ASInformationProducer.class);
    private static final String topicPrefix = "00815-";
    private final KafkaTemplate<String, String> kafkaTemplate;
```

```
    private final ObjectMapper objectMapper;


    public ASInformationProducer(KafkaTemplate<String, String>
kafkaTemplate, ObjectMapper objectMapper) {
        this.kafkaTemplate = kafkaTemplate;
        this.objectMapper = objectMapper;
    }


    public void send(ASInformation asInformation) {
        try {
            kafkaTemplate.send(topicPrefix + "as-information",
objectMapper.writeValueAsString(asInformation));
        } catch (JsonProcessingException e) {
            LOGGER.error("Error occured while processing JSON.", e);
        }
    }
}
```

Für den Start eines Prozesses über das Topic „[Präfix]-as-start-complex“ (komplexe Datenstruktur am Startknoten des Prozesses) gibt es keinen Unterschied des JSON-String zum Aufruf über die REST-Schnittstelle.

Beispiel:

```
{
    "kennzeichen": "Kennzeichen des Prozesses",
    "version": 0,
    "externeId": "0",
    "schutzstufe: "",
    "datenKennzeichen": "Kennzeichen der komplexen
         Datenstruktur|(latest oder Versionsnummer)",
    "daten": {}
}
```

Für den Start eines Prozesses über das Topic „[Präfix]-as-start-ped“ (PED-Definition am Startknoten des Prozesses) sieht der Aufbau des JSON-String wie folgt aus:

```
{
    "prozessKennzeichen": "Kennzeichen des Prozesses",
    "prozessmodellVersion": 1,
    "pedKennzeichen": "Kennzeichen der PED-Definition",
    "externeId": "0",
    "schutzstufe: "",
    "pedWerte": {
        <schlüssel>:<wert>,
        ...
    }
}
```

Für den Start eines Prozesses über die Topics „[Präfix]-as-start-fed“ (Fallakte|ng am Startknoten) oder „[Präfix]-as-start-bitgoantrag“ (bitGo-Antrag am Startknoten) sieht der Aufbau des JSON-String wie folgt aus:

```
{
    "kennzeichen": "Kennzeichen des Prozesses",
    "version": 1,
    "fachobjektOid": 0,
    "businessKey": "0",
    "bearbeiterId": ,
    "herkunft": "",
}
```

Die Variable „businessKey“ muss für die Kombination „kennzeichen“ und „version“ eindeutig sein. Die Variable „herkunft“ muss mit einer Code-ID aus der Codetabelle „ASProzessinstanzHerkunft“ gefüllt sein.

3 Prozessmodellierung

3.1 Aktivitäten

3.1.1 Verwendung von Sperrkennzeichen

Prozessinstanzen werden in der Aktionssteuerung parallel verarbeitet. Hierbei ist es möglich, dass in einem Prozess mehrere Prozessinstanzen Daten zu einem Versicherten verarbeiten. Dies kann dazu führen, dass bei der Verarbeitung in einer der Prozessinstanzen ein falsches Ergebnis erzeugt wird, da die Daten nacheinander hätten verarbeitet werden müssen. Dies ist speziell bei der Abrechnung von sonstigen Leistungen aufgefallen, aber auch im Bereich Zahlungsverkehr kann es zu Problemen kommen, wenn mehrere Prozessinstanzen auf dasselbe Zahlungsverkehrskonto zugreifen möchten.

Ein Automatismus, der die Konstellationen im Hintergrund prüft und steuert, ist technisch nicht möglich. Es wurde die Möglichkeit geschaffen, über entsprechende Aktivitäten Sperrkennzeichen zu setzen bzw. zu entfernen. Das Sperrkennzeichen bezieht sich auf Prozessinstanzen zu einem Prozess und blockiert nicht die Verarbeitung von Prozessinstanzen in anderen Prozessen.

Das Sperrkennzeichen kann hierbei ein Ordnungsbegriff oder auch ein Fachobjekt sein.

Es ist nicht generell erforderlich, Sperrkennzeichen in Prozessen zu verwenden. Wir empfehlen die Verwendung nur in kritischen Konstellationen. Bei Unsicherheit zu den Aktivitäten wenden Sie sich bitte an den entsprechenden Fachbereich.

3.1.1.1 Prüfen einer Sperre

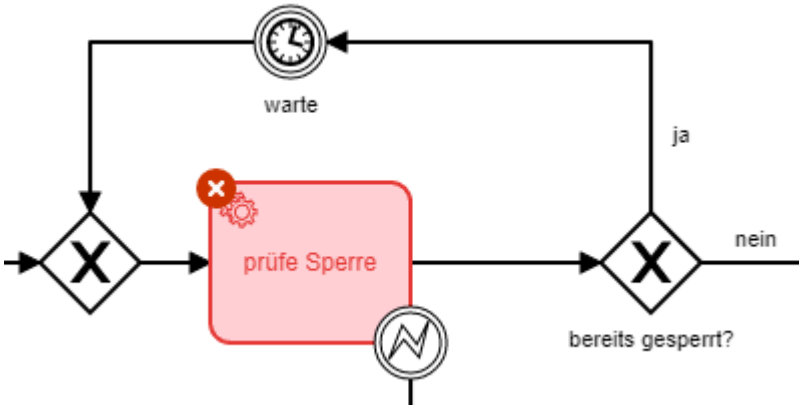
Mit der Aktivität „Prüfe und Setze Sperrkennzeichen“ kann geprüft werden, ob eine Sperre vorhanden ist.

Beispiel:

Die Aktivität wurde so modelliert, dass bis zu 4-mal auf eine vorhandene Sperre des Fachobjekts „Bonusteilnahme“ geprüft wird. Ist nach dem letzten Durchlauf weiterhin eine Sperre vorhanden, bricht die Aktivität mit dem fachlichen Fehler FE0106 ab.

Die Prüfung erfolgt NICHT in einer Schleife innerhalb der Aktivität, sondern ist entsprechend zu modellieren.

Beispiel:



Die Aktivität „Prüfe und Setze Sperrkennzeichen“ wird durchlaufen und am nachfolgenden Gateway wird die Ausgangsvariable „Ist Gesperrt“ geprüft. Liegt eine Sperre vor, wird der Pfad so modelliert, dass in diesem Fall der Prozess in ein Timer-Ereignis läuft. Die Wartezeit (Dauer) sollte hier im niedrigen Sekundenbereich liegen. Im Anschluss wird erneut die Aktivität „Prüfe und Setze Sperrkennzeichen“ durchlaufen. Ist keine Sperre vorhanden, wird der Prozess über den entsprechenden Sequenzfluss fortgesetzt.

Die Anzahl der Durchläufe der Aktivität wird über eine interne Variable festgehalten, sodass hierfür keine zusätzliche Variable benötigt wird bzw. übergeben werden muss.

Ist nach der vorgegebenen Anzahl von Durchläufen weiterhin eine Sperre vorhanden, wird der fachliche Fehler FE0106 ausgegeben und kann über ein Fehlerereignis an der Aktivität abgefangen werden.

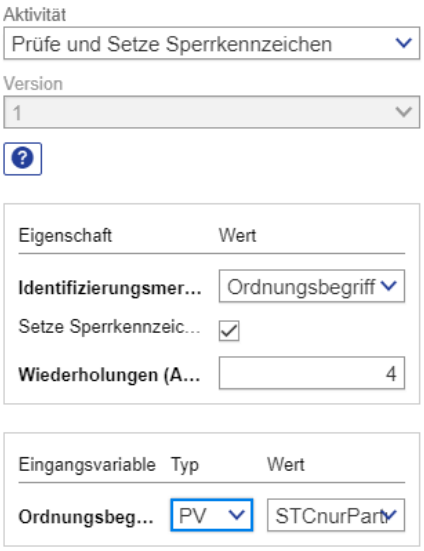
3.1.1.2 Setzen einer Sperre

Das Setzen der Sperre erfolgt mit der Aktivität „Prüfe und Setze Sperrkennzeichen“, indem das Flag „Setze Sperrkennzeichen“ gesetzt wird. Bevor die Sperre gesetzt wird, erfolgt die Prüfung, ob eine entsprechende Sperre bereits vorhanden ist. Nur wenn keine entsprechende Sperre vorhanden ist, wird diese gesetzt und der Prozess wird mit der nächsten Aktivität fortgesetzt.

Sperre eines Ordnungsbegriffs:

Ist die Verarbeitung für einen Ordnungsbegriff gesperrt, können andere Prozessinstanzen keine Daten zu diesem Ordnungsbegriff verarbeiten, solange eine Sperre aktiv ist.

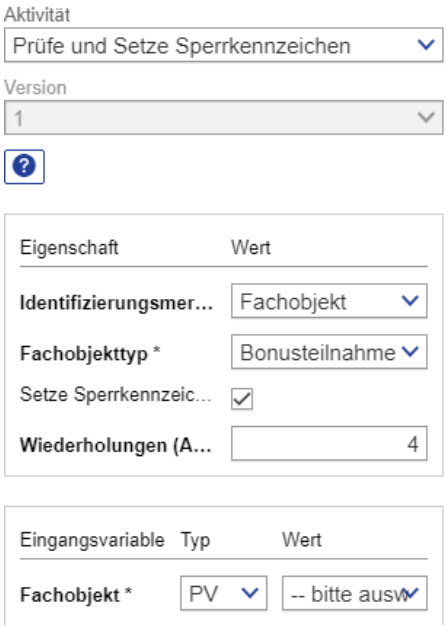
Beispiel:



Hier erfolgt die Sperre anhand einer PartnerID.
Hierdurch wird jegliche parallele Verarbeitung zu dem Partner unterbunden.

Sperre eines Fachobjekts:
Ist die Verarbeitung für ein Fachobjekt gesperrt, können andere Prozessinstanzen dieses Fachobjekt nicht verarbeiten, solange die Sperre aktiv ist.

Beispiel:



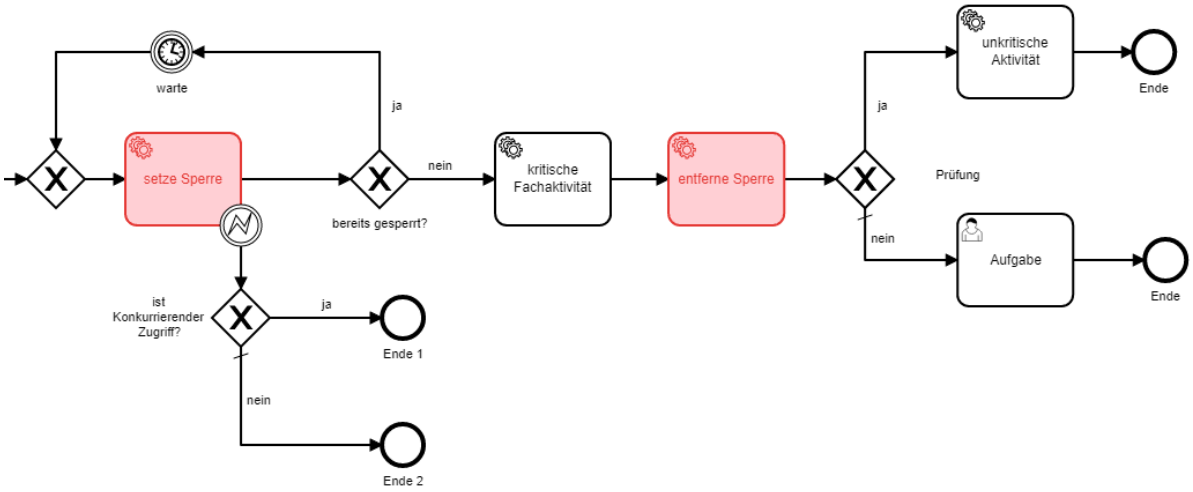
Hier wird eine Sperre auf eine konkrete Bonusteilnahme gesetzt.
Andere Prozessinstanzen können diese Bonusteilnahme nicht bearbeiten, allerdings können alle anderen Daten zu einem Partner parallel bearbeitet werden.

3.1.1.3 Entfernen einer Sperre

Das Entfernen einer Sperre erfolgt über die Aktivität „Entferne Sperrkennzeichen“. Es muss darauf geachtet werden, dass das gleiche Identifizierungsmerkmal wie beim Setzen der Sperre verwendet wird, damit die Sperre korrekt entfernt wird und andere Prozessinstanzen nicht mehr blockiert sind.

WICHTIG:
Das Entfernen eines Sperrkennzeichens sollte erfolgen, sobald die exklusive Verarbeitung abgeschlossen ist und nicht erst am Ende eines Prozesses. Zusätzlich muss darauf geachtet werden, dass das Entfernen eines Sperrkennzeichens bereits vor einem UserTask erfolgt isst. Dies hätte sonst zur Folge, dass andere Prozessinstanzen blockiert sind, solange die zugehörige Aufgabe nicht bearbeitet wurde.

In einem Prozessmodell kann das beispielsweise so aussehen:



3.1.2 Aktivität „Filtere Liste“

Ab der Version 2 bietet die Aktivität die Möglichkeit, Fachobjekte in Abhängigkeit von den Inhalten zu filtern.

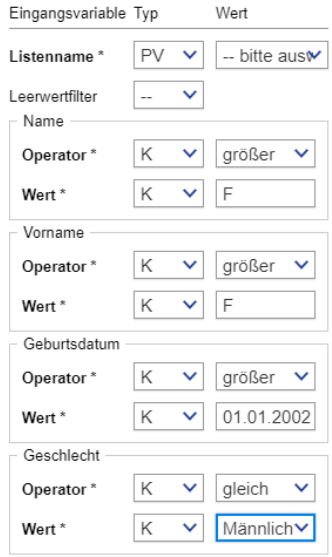
Hier am Beispiel des Fachobjekts Partner:

Einträge in den Eigenschaften

- Anzahl Filterkriterien
  Anhand der eingegebenen Anzahl wird eine entsprechende Anzahl von Schlüsseln für die Filterung aufgebaut
- Verknüpfung
  UND: Alle Kriterien müssen erfüllt sein
  ODER: mindestens eines der Kriterien muss erfüllt sein
- Fachobjekttyp:
  Auswahl des Fachobjekttyps
  Je nach Fachobjekt stehen in den Schlüsseln unterschiedliche Attribute zur Verfügung
- Schlüssel
  Die Schlüssel werden fortlaufend nummeriert, die Anzahl entspricht der Vorgabe aus „Anzahl Filterkriterien“

Eigenschaft	Wert
Anzahl Filterkriterien	4
Verknüpfung *	UND
Typ Liste *	Fachobjekt
Fachobjekttyp *	Partner
Schlüssel 1	Name
Schlüssel 2	Vorname
Schlüssel 3	Geburtsdatum
Schlüssel 4	Geschlecht

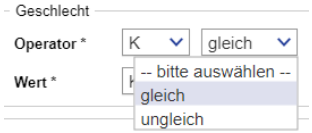
Einträge in den Eingangsvariablen



In den Eingangsvariablen ist die zu filternde Liste unter „Listenname“ einzutragen.
Für die Attribute aus den Eigenschaften wird jeweils eine Gruppe gebildet, die einen Operator und den Wert beinhaltet.

Im obigen Beispiel werden aus der Liste Partner gefiltert, deren Name und Vorname mit F-Z beginnt, nach dem 01.01.2002 geboren und männlich sind.

Je nach Variablentyp stehen nur bestimmte Operatoren zur Verfügung.

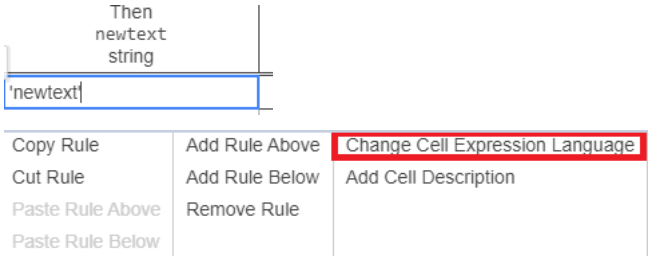


3.2 DMN-Editor

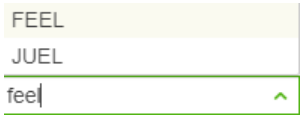
3.2.1 Verwendung von JUEL-Ausdrücken

Im DMN-Editor besteht die Möglichkeit, JUEL als Sprache auszuwählen. Mit einem JUEL-Ausdruck ist es möglich, einem Ausgangswert den Wert einer Prozessinstanzvariablen zuzuweisen.

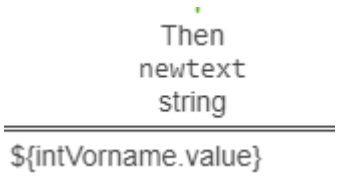
Hierzu in das gewünschte Feld klicken und mit der rechten Maustaste das Kontextmenü öffnen:



Im Kontextmenü den Eintrag „Change Cell Expression Language“ anklicken und dort den Eintrag JUEL auswählen



Um dann den Wert einer Prozessinstanzvariablen als Wert zu übernehmen ist in dem Feld die Syntax ${VARIABLENNAME.value} zu verwenden:



3.3 Verwendung von komplexen Ausdrücken (JUEL)

3.3.1 Prüfungen auf Variablen vom Typ String (Text)

- Hat die Variable einen Eintrag?
  ${PVname.value != null}
- Liegt ein bestimmter Text vor?
  ${PVname.value == „Wert“}
- Hat der Text eine bestimmte Länge?
  ${PVname.value.length() == Wert}
- Liegt an Stelle x ein bestimmtes Zeichen vor?
  ${PVname.value.charAt(Stelle) == „Wert“}
- Liegt ab Stelle x ein bestimmter Text vor?
  ${PVname.value.substring(Stelle) == „Wert“}
- Liegt von Stelle x bis y ein bestimmter Text vor?
  ${PVname.value.substring(Stelle,Stelle) == „Wert“}

3.3.2 Prüfungen auf Variablen vom Typ Date (Datum)

- Liegt ein bestimmtes Datum vor?
  ${PVname.value == dateTime().minusMonths(Wert).toDate()}

  Mögliche Parameter sind z.B.:
  minusYears()
  minusMonths()
  minusDays()
  plusYears()
  plusMonths()
  plusDays()

  Die Parameter können auch kombiniert werden wie z.B. minusMonths(2).minusDays(15)

- Handelt es sich um ein Schaltjahr?

  ${ intJahr.value % 4 == 0 && (!(intJahr.value % 100 == 0) || intJahr.value % 400 == 0)) }

  Mit diesem Ausdruck kann an einem Gateway geprüft werden, ob es sich bei einem Jahr um ein Schaltjahr handelt.

3.3.3 Prüfungen für Listen

- Ist eine Liste leer?
  ${PVlistenname.list.isEmpty()}

  Die Prüfung ist nur möglich, wenn die Listenvariable vorhanden ist, also PVlistenname.list != null
- Wie viele Einträge hat eine Liste?
  ${PVlistenname.list.size() == Wert}

3.3.4 Ausdrücke für Variablen vom Typ String (Text)

- Umwandlung in Kleinbuchstaben
  ${execution.getVariable('PVname').setValueTyped(execution.getVariable('PVname').value.toLowerCase())}
- Kürzung auf die letzten x Stellen
  ${execution.getVariable('PVname').setValueTyped(PVname.value.substring(PVname.value.length() -Wert))}

  Die Funktionalität kann mit der Aktivität "Extrahiere Text" abgebildet werden. Hierzu ist in den Eigenschaften das Flag "letzte n Stellen ermitteln" zu setzen. In den Eingangsvariablen wird die Stringvariable mitgegeben sowie die Anzahl der Zeichen, die vom Ende des Strings genommen werden sollen. Das Ergebnis wird in die Ausgangsvariable geschrieben.
- Kürzung von Stelle x bis y
  ${execution.getVariable('PVname').setValueTyped(PVname.value.substring(Stelle1,Stelle2))}

  Die Funktionalität kann mit der Aktivität "Extrahiere Text" abgebildet werden. Hierzu ist in den Eigenschaften das Flag "letzte n Stellen ermitteln" nicht zu setzen. In den Eingangsvariablen wird die Stringvariable mitgegeben, sowie die Startposition und die Anzahl der Zeichen, die von der Startposition des Strings berücksichtigt werden sollen. So können auch die ersten x Zeichen extrahiert werden.

  Das Ergebnis wird in die Ausgangsvariable geschrieben.

- Verbinden von Variablen
  ${execution.getVariable('PVname1').setValueTyped(execution.getVariable('PVname1').value.concat(' ').concat(execution.getVariable('PVname2').value.concat(,Wert')).concat(execution.getVariable('PVname3').value))}

  Die Funktionalität kann mit der Aktivität "Konkateniere Werte" abgebildet werden. In den Eigenschaften ist anzugeben wie viele Variablen verkettet werden sollen.

In den Eingangsvariablen wird der Inhalt entsprechend aufgebaut.
Je Eingangsvariable wird der Wert mitgegeben (PV, SV, K) und je Variable kann ein Trennzeichen (Suffix) gesetzt werden. Mit der Option "Entferne Leerzeichen" werden alle Leerzeichen in dem übergebenen String entfernt.
Mit der Aktivität kann man aus einem Text auch nur die Leerzeichen entfernen, indem man bei "Anzahl Werte" 1 einträgt und "Entferne Leerzeichen" auswählt.

- Variablen mit einem Wert überschreiben
  ${execution.getVariable('PVname').setValueTyped('Wert‘)}
  Bei Zahlen ist der Wert ohne Hochkomma zu setzen.

  Die Funktionalität kann mit der Aktivität "Aktualisiere Variable" abgebildet werden. Hier wird eine bestehende Variable mit "Neuer Wert" überschrieben, daher besitzt die Aktivität keine Ausgangsvariable.
- Text in Variable ersetzen
  ${<PVvariable>.value != null ?
  <PVvariable>.setValueTyped(<PVvariable>.value.replace('tonline.de','t-online.de')) :
  <PVvariable>.value}

  Hier wird zunächst geprüft, ob die Variable gefüllt ist. Wenn das der Fall ist, wird in dem Text 'tonline' durch 't-online' ersetzt, ansonsten bleibt die Variable unverändert.
- Leerzeichen in Stringvariable am Anfang und am Ende entfernen
  ${adresseVariable.setValueTyped(adresseVariable.value.trim())}

3.3.5 Ausdrücke für Variablen vom Typ Date (Datum)

- Tagesdatum um einen Tag erhöhen
  ${PVname.value.setTime(PVname.value.getTime() + (1000 * 60 * 60 * 24))}

  Die Funktionalität kann mit der Aktivität "Berechne Datum" abgebildet werden. In den Eingangsvariablen wird das Ausgangsdatum übergeben, der Operator (+/-) und die Anzahl von Tagen.

3.3.6 Ausdrücke für Variablen vom Typ Integer (Zahl)

- Zähler um einen Wert erhöhen
  ${execution.getVariable('PVname').setValueTyped(PVname.value +Wert)}

  Das Inkrement ist so in der Form nicht direkt über Aktivitäten abbildbar.
  Alternative:
  1. Aktivität "Erstelle Variable" vom Typ Dezimal mit dem Wert 1,00.
  2. Aktivität "Berechne Betrag" mit Betrag 1 = Variable aus 1. und Addition von Betrag 2 = "K 1,00"

  Am Gateway lässt sich die Variable dann prüfen, indem man den Dezimalwert angibt, z.B. 3,00.

Auf diese Weise kann man Schleifen bauen, die mehrmals durchlaufen werden sollen, bis zu einer bestimmten Maximalzahl.

3.4 Gateways

3.4.1 Prüfung auf 0 bei Dezimalzahlen

Leider funktioniert der Operator "gleich 0" nicht wie erwartet mit dem Bigdecimal. Allerdings funktionieren die Operatoren "kleiner gleich 0" und "größer gleich 0" wie erwartet mit dem Bigdecimal. Daher muss man für einen 0-Check diese beiden Operatoren in einem Pfad des Gateways mit einem „UND“-Operator verknüpfen. Damit ist dann sichergestellt, dass die überprüfte Variable auch wirklich den Wert 0 hat.

4 Verfügbare Standardvariablen

Im Folgenden werden die Standardvariablen erläutert, die für Eingangsvariablen in Service-Tasks genutzt werden können.

- HEUTE

  Verfügbar an allen Eingangsvariablen vom Typ Date oder Datetime. Bei Auswahl wird in die Variable das zur Laufzeit aktuelle Tagesdatum als Wert eingetragen.

- Prozessinstanz-ID

  Verfügbar für alle Eingangsvariablen vom Typ String, sofern die Aktivitäten dies in der Implementierung zulassen. Bei Auswahl wird die Eingangsvariable mit der Prozessinstanz-ID befüllt.