Customizing the Order Fulfillment Process

The purpose of this trail is to demonstrate how to customize the order fulfillment process in our Telcotraining shop.

Story:

Let's assume that we want to implement an immediate cancel mechanism in our Telcotraining store.
An immediate cancel would allow the customer to cancel an order immediately after placing it.
Orders would be eligible for immediate cancel depending on the amount of time that has passed since the order placement. The amount of time should be configurable.
To implement this feature, we need to :
1.) Modify the fulfillment process. We need to introduce a waiting node in our fulfillment pipeline that waits a certain amount of time before the order is fulfilled.
2.) Customize the Order Cancel service to make pending orders eligible for immediate cancellation.
We will address the first of these in this trail, and the second in the following trail.

Modify the fulfillment process

Review the existing fulfillment process

Please have a look at how the fulfillment process is defined: **`Telcotrainingfulfilmentprocess/resources/Telcotrainingfulfilmentprocess/process/order-process.xml`** It is a process definition using the process engine framework, which is part of the processengine module.
Note how the order is processed by a pipeline of consecutive nodes (ACTIONS).
These node are carefully ordered: order verification, fraud checking, payment, ..

Note also that the order is split into consignments (the `splitOrderAction`); internally, this action fires a new process (**`Telcotrainingfulfilment/resources/Telcotrainingfulfilmentprocess/process/consignment-process.xml`**) for each consignment, handling the warehouse integration for a particular consignment fulfillment.

The order-process pipeline then waits for each subprocess to end, with the `subprocessesCompletedAction` looping back to the wait node until all subprocesses are finished executing.

What we need to implement is an action, which we will put at the start of the pipeline, that will put an order on hold before it is pushed to the first fulfillment action.

Add New Order Status

Normally, each order fulfillment action changes the state of the order to indicate its progress along the pipeline. Since we want to add a new fulfillment node, it's a good idea to provide a new order state indicating that the order is being processed by our node in the pipeline.

Add a new order status enum value in **`Telcotrainingfulfilmentprocess-items.xml`**. Our action node will set the status of the order to this value:

Telcotrainingfulfilmentprocess/resources/Telcotrainingfulfilmentprocess-items.xml

```
<enumtypes>
    <enumtype code="OrderStatus"
autocreate="false" generate="false" dynamic="true">
        <value code="WAITING_FOR_IMMEDIATE_CANCEL"
/>
    </enumtype>
</enumtypes>
```

Localize this new enum values in **Telcotrainingfulfilmentprocess-locales_en.properties**

Telcotrainingfulfilmentprocess/resources/localization/Telcotrainingfulfilmentprocess-locales_en.properties

```
type.OrderStatus.WAITING_FOR_IMMEDIATE_CANCEL.name
=Waiting For Immediate Cancel
```

Call **ant all** to generate the new enum value in *de.hybris.platform.core.enums.OrderStatus* .

Define 'waitForImmediateCancel' node as the first node.

Add our action's node definition to the place order process, and modify the process definition's 'start' attribute to be our new node.

Telcotrainingfulfilmentprocess/resources/Telcotrainingfulfilmentprocess/process/order-process.xml

```
<?xml version="1.0" encoding="utf-8"?>
    <action id="waitForImmediateCancel"
bean="waitForImmediateCancel">
        <transition name="OK" to="checkOrder"/>
        <transition name="NOK"
to="waitForImmediateCancel"/>
        <transition name="CANCELLED" to="failed"/>
    </action>
..
```

It has three transitions :

- to itself, if order still needs to wait for immediate cancel trigger,
- to the 'checkOrder' action which starts fulfillment, if the immediate order cancel time window has passed
- to the end of the process if the order was cancelled in the mean time.

Implement the 'waitForImmediateCancel' node class:

Extend the AbstractAction class with WaitForImmediateCancel.

Its **execute** method must perform the following tasks:

a.) check if the time that has passed since creation of the order is less than the configured *order pending* time

b.) if yes, change the order status to **WAITING_FOR_IMMEDIATE_CANCEL** and return with a NOK transition (which will cause the node to loop).

c.) if no, return with OK transition (continue to the next node -- checkOrder in this case).

d.) if the order was cancelled or the cancelling process has already started, quit the fulfillment process.

The configured pending time can be fetched from the OrderCancelConfiguration instance, persisted in the data base. To avoid the frequent node re-entrance, put the thread to sleep before returning a NOK transition.

Telcotrainingfulfilmentprocess/src/de/hybris/Telcotraining/fulfilmentprocess/actions/order/WaitForImmediateCancel.java Collapse source

```
package de.hybris.Telcotraining.fulfilmentprocess.actions.order;
import de.hybris.platform.core.enums.OrderStatus;
import de.hybris.platform.core.model.order.OrderModel;
import de.hybris.platform.ordercancel.dao.OrderCancelDao;
import de.hybris.platform.ordercancel.model.OrderCancelConfigModel;
import de.hybris.platform.orderprocessing.model.OrderProcessModel;
```

```
import
de.hybris.platform.processengine.action.AbstractAc
tion;
import
de.hybris.platform.processengine.model.BusinessPro
cessModel;
import
de.hybris.platform.task.RetryLaterException;
import java.util.Date;
import java.util.Set;
import javax.annotation.Resource;
import org.apache.log4j.Logger;
public class WaitForImmediateCancel extends
AbstractAction
{
    private static final String STATUS_NOK = "NOK";
    private static final String STATUS_OK = "OK";
    private static final String STATUS_CANCELLED =
"CANCELLED";
    private static Set<String> transitions =
createTransitions(STATUS_OK, STATUS_NOK);
    private final static Logger LOG =
Logger.getLogger(WaitForImmediateCancel.class);
    private OrderCancelDao orderCancelDao;
    @Override
    public String execute(final
BusinessProcessModel process) throws
RetryLaterException, Exception
    {
        if (process instanceof OrderProcessModel)
        {
            final OrderProcessModel orderProcess =
(OrderProcessModel) process;
            final OrderModel order =
orderProcess.getOrder();
            getModelService().refresh(order);
```

```
            if
(OrderStatus.CANCELLED.equals(order.getStatus())
||
OrderStatus.CANCELLING.equals(order.getStatus()))
            {
                LOG.debug("Process [" +
process.getCode() + "] : Order [" +
order.getCode() + "] was cancelled");
                return STATUS_CANCELLED;
            }
            if (stillWait(order))
            {
                if (LOG.isDebugEnabled())
                {
                    LOG.debug("Process [" +
process.getCode() + "] : Order [" +
order.getCode() + "] in a wait node");
                }
                updateStatus(order);
                try
                {
                    Thread.sleep(2000);
                }
                catch (final InterruptedException
e)
                {
                    LOG.error("Sleep interrupted",
e);
                }
                return STATUS_NOK;
            }
            return STATUS_OK;
        }
        throw new IllegalStateException("Process
must be of type " +
OrderProcessModel.class.getName());
    }
```

```
    private void updateStatus(final OrderModel
order)
    {
        getModelService().refresh(order);
        if (!
OrderStatus.WAITING_FOR_IMMEDIATE_CANCEL.equals(or
der.getStatus()))
        {
            order.setStatus(OrderStatus.WAITING_FO
R_IMMEDIATE_CANCEL);
            getModelService().save(order);
        }
    }
    @Override
    public Set getTransitions()
    {
        return transitions;
    }
    private boolean stillWait(final OrderModel
order)
    {
        final long orderPlacedMilis =
order.getCreationtime().getTime();
        final long timetowait =
getImmediateCancelWaitingTime() * 1000;
        return (new Date().getTime() <
(orderPlacedMilis + timetowait));
    }
    private int getImmediateCancelWaitingTime()
    {
        return
getCancelConfiguration().getQueuedOrderWaitingTime
();
    }
    private OrderCancelConfigModel
getCancelConfiguration()
    {
```

```
        return
getOrderCancelDao().getOrderCancelConfiguration();
    }
    @Resource
    public void setOrderCancelDao(final
OrderCancelDao orderCancelDao)
    {
        this.orderCancelDao = orderCancelDao;
    }
    /**
     * @return the orderCancelDao
     */
    protected OrderCancelDao getOrderCancelDao()
    {
        return orderCancelDao;
    }
}
```

Add spring configuration Telcotrainingfulfilmentprocess/resources/ Telcotrainingfulfilmentprocess/process/order-process-spring.xml

```
...
<bean id="waitForImmediateCancel"
class="de.hybris.Telcotraining.fulfilmentprocess.a
ctions.order.WaitForImmediateCancel"
parent="abstractAction"/>
...
```

Localize display status

Add mapping

Telcotrainingfulfilmentprocess/resources/ Telcotrainingfulfilmentprocess-spring.xml

```
<alias alias="dynamicAttributesOrderStatusDisplay"
name
="TelcotrainingDynamicAttributesOrderStatusDisplay
ByMap"/>
<bean
id="TelcotrainingDynamicAttributesOrderStatusDispl
```

```
ayByMap"
parent="acceleratorDynamicAttributesOrderStatusDis
playByMap">
    <property name="statusDisplayMap">
        <map merge="true">
            <entry
key="WAITING_FOR_IMMEDIATE_CANCEL"
value="waitingForImmediateCancel"/>
        </map>
    </property>
</bean>
```

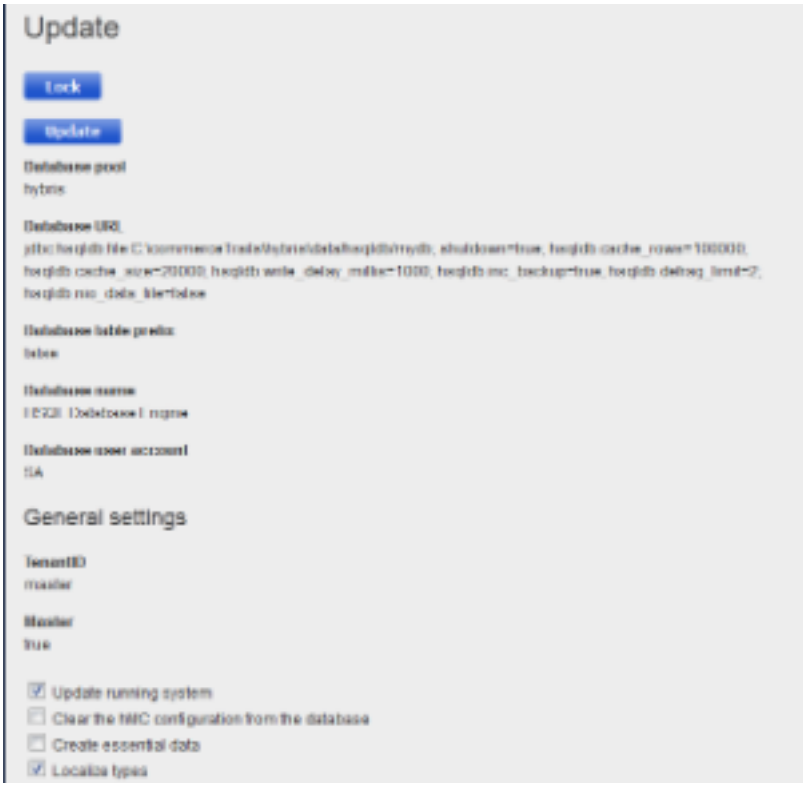
Telcotrainingstorefront/web/webroot/WEB-INF/messages/base_en.properties

```
text.account.order.status.display.waitingForImmedi
ateCancel = Waiting For Immediate Cancel
```

Rebuild and update

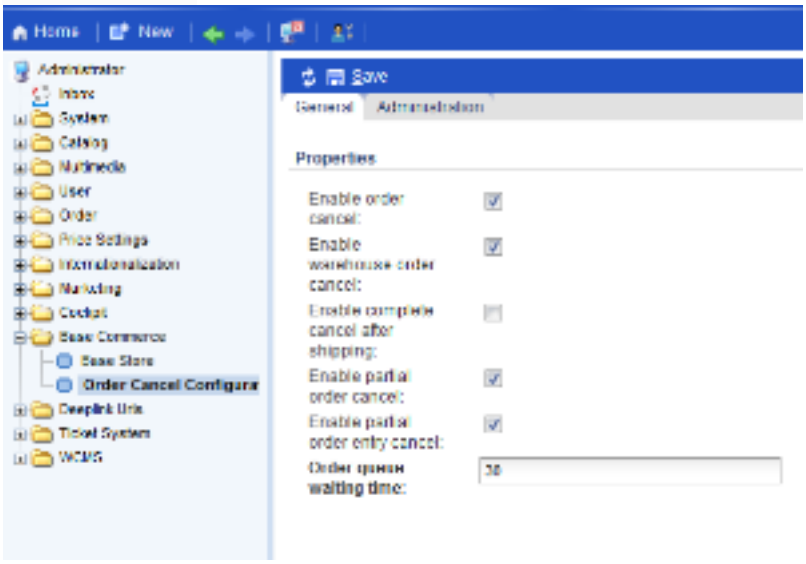
Rebuild the code base. Call **ant all** and restart the hybris server. As we introduced a new enumtype, a system update is needed.

Go to http://localhost:9001/platform/update and run system update with type localization enabled. No project data initialization is required.



Verify the new fulfillment action:

Go to hmc and configure the order queuing time to 30 seconds:



Go to the Telcotraining shop and place an order. Verify on the customer's orders page that the status of your new order stays in pending state for 30 seconds:

After 30 seconds your order should be pushed to the fulfillment pipeline and the status should be changed: