File - C:\dev\Project\test-sau\ptp-sau-tests\src\main\java\com\nasdaq\nv\testcases\functional\clearing\settlement\TcSettlement.java

```java
package com.nasdaq.nv.testcases.functional.clearing.settlement;

import com.nasdaq.api.ptp.se.messages.SB10007;
import com.nasdaq.nv.datamodel.factory.TradeDataFactory;
import com.nasdaq.nv.framework.Collector;
import com.nasdaq.nv.framework.annotation.Framework;
import com.nasdaq.nv.functionarea.cl.TradeFunctionsNew;
import com.nasdaq.nv.functionarea.se.SettlementFunctions;
import com.nasdaq.nv.message.collection.CB10001Collection;
import com.nasdaq.nv.messages.implapi.se.SB10007Impl;
import com.nasdaq.nv.messages.rdm.structure.NvInstSeries;
import com.nasdaq.nv.nvs.session.NvsMessageUtil;
import com.nasdaq.nv.testcases.system.BatchLogChecker;
import om.omexus.batchrunner.main.ExecutionOrder;
import om.omexus.batchrunner.model.FunctionalTestCollection;
import om.omexus.commons.date.BusinessDateUtil;
import om.omexus.functionarea.account.AccountUtil;
import om.omexus.functionarea.annotations.setup.AnnotationSupport;
import om.omexus.functionarea.annotations.setup.TCParameterProp;
import om.omexus.functionarea.productData.tradedata.TradeData;
import om.omexus.functionarea.productData.tradedata.TradeToolDelux;
import om.omexus.functionarea.scenario.action.transaction.trade.structure.TradeItem;
import om.omexus.message.cm.constants.CollateralTxState;
import om.omexus.omex.omnet.OMexusException;
import om.omexus.shared.log.OMexusLog;
import om.omexus.testcase.TestException;
import om.omexus.testcases.CoreTestCase;
import om.omexus.toolbox.cdb.CDBSeriesInfo;
import om.omexus.toolbox.util.SystemInfoUtil;
import om.omnet.structure.account_t;
import org.assertj.core.api.Assertions;
```

```java
import org.junit.jupiter.api.Test;

import java.util.List;

//import static com.nasdaq.nv.testcases.functional.clearing.rm.util.CCARUtil.
assertDeltaHedgeIsMarginModel;
import static com.nasdaq.nv.testcases.functional.clearing.rm.util.DeltaHedgeUtil.getNvsExUser;
import static com.nasdaq.nv.testcases.functional.clearing.rm.util.DeltaHedgeUtil.
shouldUseNddBroadcast;
//import static com.nasdaq.nv.testcases.functional.clearing.rm.util.TotalMarginCalculationUtil.
setSettlementPosition;
//import static com.nasdaq.nv.testcases.functional.clearing.rm.util.VariationMarginCalculationUtil.
getNvInstSeries;
import static om.omexus.toolbox.util.SessionHandler.getNvsSession;


/**
 * @author marjab & valrim
 * @abstract ----    ----
 * • Enable/Disable customized total margin formula
 * • Identify different set of settlement positions
 * • Calculate Initial and Variation Margin
 * • Compare results together with Minimum Margin Requirement
 * • Determine Total Margin Requirement
 * @action
 * @weak_points
 * @result
 * @assumption
 * @reference
 * @since
 */
```

File - C:\dev\Project\test-sau\ptp-sau-tests\src\main\java\com\nasdaq\nv\testcases\functional\clearing\settlement\TcSettlement.java

```
59 @FunctionalTestCollection(
60         description = "APP-CM-POC test",
61         userNameDescriptions = {"Dummy"},
62         autoLogin = false
63 )
64 public class TcSettlement extends CoreTestCase implements AnnotationSupport {
65
66     private static final int MILL_SEC_TO_WAIT = 5_000;
67     private static final String BATCH_STATE = "COMPLETED";
68     private static final BatchLogChecker BATCH_LOG_CHECKER = new BatchLogChecker();
69
70     @TCParameterProp("tradeSource")
71     private int tradeSource;
72
73     @Framework
74     private TradeDataFactory tradeDataFactory;
75     @Framework
76     private SettlementFunctions settlementFunctions;
77     @Framework
78     private TradeFunctionsNew tradeFunctions;
79     @Framework
80     private Collector collector2;
81
82
83     private account_t account1;
84     private account_t account2;
85     private NvInstSeries nvInstSeries;
86     private NvInstSeries nvInstSeries2;
87     private NvInstSeries nvInstSeries3;
88     private NvInstSeries nvInstSeries4;
89     private NvInstSeries nvInstSeries5;
```

```
 90     private String businessDate;
 91     private CDBSeriesInfo seriesCdb;
 92     private CDBSeriesInfo seriesCdb2;
 93     private CDBSeriesInfo seriesCdb3;
 94     private CDBSeriesInfo seriesCdb4;
 95     private CDBSeriesInfo seriesCdb5;
 96     private String trader1;
 97     private String trader2;
 98     private TradeToolDelux tradeTool;
 99     private final OMexusLog log = OMexusLog.getInstance();
100
101     @Override
102     public void tcSetup() throws OMexusException {
103         super.tcSetup();
104
105         iNvsSession = getNvsSession();
106         NvsMessageUtil.sendBroadcastSubcription(iNvsSession, "EB518");
107         NvsMessageUtil.sendBroadcastSubcription(iNvsSession, "JB257");
108         NvsMessageUtil.sendBroadcastSubcription(iNvsSession, "JB10005");
109         NvsMessageUtil.sendBroadcastSubcription(iNvsSession, "JB10009");
110         NvsMessageUtil.sendBroadcastSubcription(iNvsSession, "SB1");
111         NvsMessageUtil.sendBroadcastSubcription(iNvsSession, "SB10007");
112
113
114
115         iNvsSession.getConnection().startListenToBc();
116         businessDate = SystemInfoUtil.getBusinessDate();
117
118         tradeTool = new TradeToolDelux(iNvsSession, businessDate, SystemInfoUtil.
    getSystemDateAndTime(), shouldUseNddBroadcast(iOmexusConfiguration));
119         tradeTool.setExUserCode(getNvsExUser(iOmexusConfiguration)[0], getNvsExUser(
```

```
119 iOmexusConfiguration)[1], getNvsExUser(iOmexusConfiguration)[2]);
120         tradeTool.setPlaceOfSettlementId(1);
121         tradeTool.setWaitTime(MILL_SEC_TO_WAIT);
122         trader1 = iOmexusConfiguration.getTestCaseParam("trader1");
123         trader2 = iOmexusConfiguration.getTestCaseParam("trader2");
124 
125         seriesCdb = getReferenceDataToolbox().getSeriesByName(iOmexusConfiguration.getTestCaseParam(
    "iSeries_1_CDB"));
126         seriesCdb2 = getReferenceDataToolbox().getSeriesByName(iOmexusConfiguration.getTestCaseParam
    ("iSeries_2_CDB"));
127         seriesCdb3 = getReferenceDataToolbox().getSeriesByName(iOmexusConfiguration.getTestCaseParam
    ("iSeries_3_CDB"));
128         seriesCdb4 = getReferenceDataToolbox().getSeriesByName(iOmexusConfiguration.getTestCaseParam
    ("iSeries_4_CDB"));
129         seriesCdb5 = getReferenceDataToolbox().getSeriesByName(iOmexusConfiguration.getTestCaseParam
    ("iSeries_5_CDB"));
130 
131         account1 = AccountUtil.asAccount(iOmexusConfiguration.getTestCaseParam("account1"));
132         account2 = AccountUtil.asAccount(iOmexusConfiguration.getTestCaseParam("account2"));
133     }
134 
135 
136     private TradeData enterTrade(CDBSeriesInfo series, int quantity, double price, account_t
    buyAccount, account_t sellAccount, String freeText) throws OMexusException {
137         int adjPrice = (int) (price * Math.pow(10, series.getPriceDec()));
138         TradeData tradeData = tradeDataFactory.create(series, quantity, adjPrice)
139                 .buyAccount(buyAccount)
140                 .sellAccount(sellAccount)
141                 .tradeSource((short) 11)
142                 .freeText(freeText);
143         CB10001Collection cb10001Collection = tradeFunctions.enterTrade(tradeData);
```

File - C:\dev\Project\test-sau\ptp-sau-tests\src\main\java\com\nasdaq\nv\testcases\functional\clearing\settlement\TcSettlement.java

```
144         return tradeData;
145     }
146     /**
147      * @throws OMexusException
148      * @purpose for TO indicator true
149      * @author marjab xxxxxxxxScenario 3 Test case 1xxxxxxx
150      */
151     @ExecutionOrder(1)
152     @Test
153     public void testChkRealTimeDeliverySecurityIndicatorsWithTOIndicatorTrue() throws
    OMexusException {
154         //Creating the trade data with buy and sell account
155         TradeData tradeData1 = tradeTool.create().buyerInfo(trader2).buyAccount(account2).sellerInfo
    (trader1).sellAccount(account1);
156         tradeData1.tradeSource((short) 0);
157         // Entering the trade with today's Business date as Settlement trade date (T+0)
158         TradeItem tradeItem1 = tradeTool.enterTrade(seriesCdb, 100, 100_000, tradeData1.
    settlementDate(businessDate));
159 
160         getSshSession().execCommand("am rm_rts reg intraday");
161 
162         List<Object> messages =  iNvsSession.getConnection().getBroadcastFetcher().getBroadcastList(
    "SB10007", MILL_SEC_TO_WAIT);
163 
164         log.message("messages :: " + messages.toString());
165 
166         // ========Verify Trade Date and Settlement Instruction Date is same for this Settlement
    =========
167         verifyTradeDateAndSettlementInstructionDateIsSame(messages);
168 
169         // ========Verify DVPItemProperties should be there in Settlement instruction=========
```

```
170         verifyDVPItemPropertiesAvailable(messages);
171 
172         // ========Verify DVPItemProperties should be there in Instruction Reference Field=========
173         verifyInstructionReferenceFieldAvailable(messages);
174 
175 
176     }
177 
178     /**
179      * @throws OMexusException
180      * @purpose for TO indicator false
181      * @author marjab xxxxxxxxScenario 3 Test case 1xxxxxxx
182      */
183     @ExecutionOrder(2)
184     @Test
185     public void testChkRealTimeDeliverySecurityIndicatorsWithTOIndicatorFalse() throws
    OMexusException {
186         TradeData tradeData1 = tradeTool.create().buyerInfo(trader2).buyAccount(account2).sellerInfo
    (trader1).sellAccount(account1);
187         tradeData1.tradeSource((short) 0);
188         // This is T+0 Settlement trade
189         TradeItem tradeItem1 = tradeTool.enterTrade(seriesCdb, 100, 100_000, tradeData1.
    settlementDate(BusinessDateUtil.rollDate(businessDate, 2)));
190 
191         getSshSession().execCommand("am rm_rts reg intraday");
192 
193         List<Object> messages =  iNvsSession.getConnection().getBroadcastFetcher().getBroadcastList(
    "SB10007", MILL_SEC_TO_WAIT);
194 
195         // ========Verify DVPItemProperties should be there in Settlement instruction=========
196         verifyDVPItemPropertiesAvailable(messages);
```

File - C:\dev\Project\test-sau\ptp-sau-tests\src\main\java\com\nasdaq\nv\testcases\functional\clearing\settlement\TcSettlement.java

```
197
198         // ========Verify DVPItemProperties should be there in Instruction Reference Field=========
199         verifyInstructionReferenceFieldAvailable(messages);
200
201         // ========Verify Trade Date and Settlement Date should not be same for this Settlement
    =========
202         verifyTradeDateAndSettlementDateShouldNotBeSame(messages);
203     }
204
205     /**
206      * @purpose Verify DVP Item Property is available in SB10007 response for trade Settlement
207      *
208      */
209     private void verifyDVPItemPropertiesAvailable(List<Object> settlement)  throws OMexusException{
210         Assertions.assertThat(settlement.size()).isGreaterThan(0);
211
212         SB10007Impl sb10007 = (SB10007Impl)settlement.stream().map(SB10007Impl.class::cast).filter((
    set) -> {
213             return this.hasDvpItemProperties(set);
214         }).findFirst().orElseThrow(() -> {
215             return new TestException("trade Settlement does not have the expected DVPItemProperties
     ");
216         });
217     }
218
219     private boolean hasDvpItemProperties(SB10007Impl sb10007impl){
220         SB10007 sb10007 = sb10007impl.getMessage();
221         return !String.valueOf(sb10007.getItems().get(0).getDvpItemProperties()).isEmpty() &&
    sb10007.getItems().get(0).getDvpItemProperties()>0;
222     }
223
```

File - C:\dev\Project\test-sau\ptp-sau-tests\src\main\java\com\nasdaq\nv\testcases\functional\clearing\settlement\TcSettlement.java

```
224     /**
225      * @purpose Verify Instruction Reference field should be available in SB10007 response for trade
     Settlement
226      *
227      */
228     private void verifyInstructionReferenceFieldAvailable(List<Object> settlement)  throws
    OMexusException{
229         Assertions.assertThat(settlement.size()).isGreaterThan(0);
230         SB10007Impl sb10007 = (SB10007Impl)settlement.stream().map(SB10007Impl.class::cast).filter((
    set) -> {
231             return this.hasInstructionReferenceField(set);
232         }).findFirst().orElseThrow(() -> {
233             return new TestException("trade Settlement does not have the expected Instruction
    Reference Field ");
234         });
235     }
236
237     private boolean hasInstructionReferenceField(SB10007Impl sb10007impl){
238         SB10007 sb10007 = sb10007impl.getMessage();
239         return sb10007.getItems().get(0).getInstructionReference() == null;
240     }
241
242     /**
243      * @purpose verify TradeDate And Settlement Instruction Date is same is available in SB10007
    response for trade Settlement
244      *
245      */
246     public void verifyTradeDateAndSettlementInstructionDateIsSame(List<Object> settlement) throws
    OMexusException {
247         Assertions.assertThat(settlement.size()).isGreaterThan(0);
248         SB10007 sb10007  = ((SB10007Impl)settlement.get(0)).getMessage();
```

```
249         Assertions.assertThat(sb10007.getItems().get(0).getTradeDate()).isEqualTo(sb10007.getItems
    ().get(0).getSettlementInstrDate());
250     }
251 
252     /**
253      * @purpose verify TradeDate And Settlement Instruction Date is same is available in SB10007
    response for trade Settlement
254      *
255      */
256     public void verifyTradeDateAndSettlementDateShouldNotBeSame(List<Object> settlement) throws
    OMexusException {
257         Assertions.assertThat(settlement.size()).isGreaterThan(0);
258         SB10007 sb10007  = ((SB10007Impl)settlement.get(0)).getMessage();
259         Assertions.assertThat(sb10007.getItems().get(0).getTradeDate()).isNotEqualTo(sb10007.
    getItems().get(0).getSettlementDate());
260     }
261 
262     /**
263      * @purpose verify External Trade id should be available in SB10007 response for trade
    Settlement
264      *
265      */
266     public void verifyExternalTradeIdAvailable(List<Object> settlement) throws OMexusException {
267         Assertions.assertThat(settlement.size()).isGreaterThan(0);
268         SB10007 sb10007  = ((SB10007Impl)settlement.get(0)).getMessage();
269         Assertions.assertThat(sb10007.getItems().get(0).getExternalTradeId().isEmpty()).isFalse();
270     }
271 }
272 
```