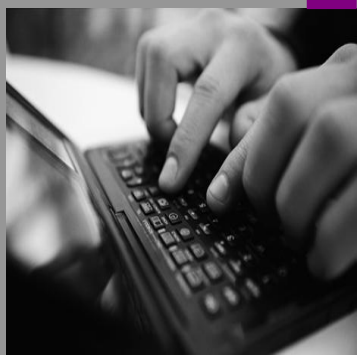


How-to Guide
SAP Mobile Business Solutions
How-To Integration xMTT and xMSA

# How To Integrate xMTT and MSA Laptop

Version <1.07> – 07 2007

Applicable Releases:
MSA Laptop 5.0
xMTT 2.0

SAP Library document classification: PUBLIC

# 1 Scenario

You want to have integration between the Mobile Travel Expenses and Mobile Sales for Laptop applications on the client.

# 2 Introduction

This document explains how you can create the integration between Mobile Travel Expenses and Mobile Sales for Laptop on the client.
The integration is split in two parts, a part in Mobile Sales for Laptop and a part in Mobile Travel Expenses.

## 2.1 Mobile Sales for Laptop Integration

In Mobile Sales for Laptop, the integration is a link from within the Mobile Sales for Laptop template to the Mobile Infrastructure client where the applications are installed.

This logic is built into the standard application. Please refer to the following notes among the available MSA-notes if you encounter any problems and for up-to-date information:

- 1001013 – If you open tiles of Time && Travel "http://localhost:4444/me/run/EXPENSE/!" or "http://localhost:4444/me/run/CATS_NOTEBOOK/!" appears
- 970254 – If you run into trouble with sync: "Data Synchronization; Send and Import fails"

NOTE: This security enhancement is valid for all SAP CRM Mobile Sales applications such as Mobile Sales (Standard - MSA), Mobile Sales for Consumer Products (MSC) and Mobile Sales for Pharmaceuticals (MSP).

## 2.2 Mobile Travel Expenses Integration

In Mobile Travel Expenses, the integration is the fetching and displaying of data from Mobile Sales for Laptop by the Mobile Travel Expenses application.

This logic is built as a customer enhancement to the standard application using the Mobile Cross Application Framework (mCAF)[1].

The enhancement logic contains SQL queries to fetch data from the Mobile Sales for Laptop application for use in the Mobile Travel Expenses application. The implementation provided is a sample integration scenario. Varieties of this logic can be implemented to provide specific integration scenarios to meet different business requirements.

Please refer to the Step-by-Step Solution section for detailed steps on implementing the integration.

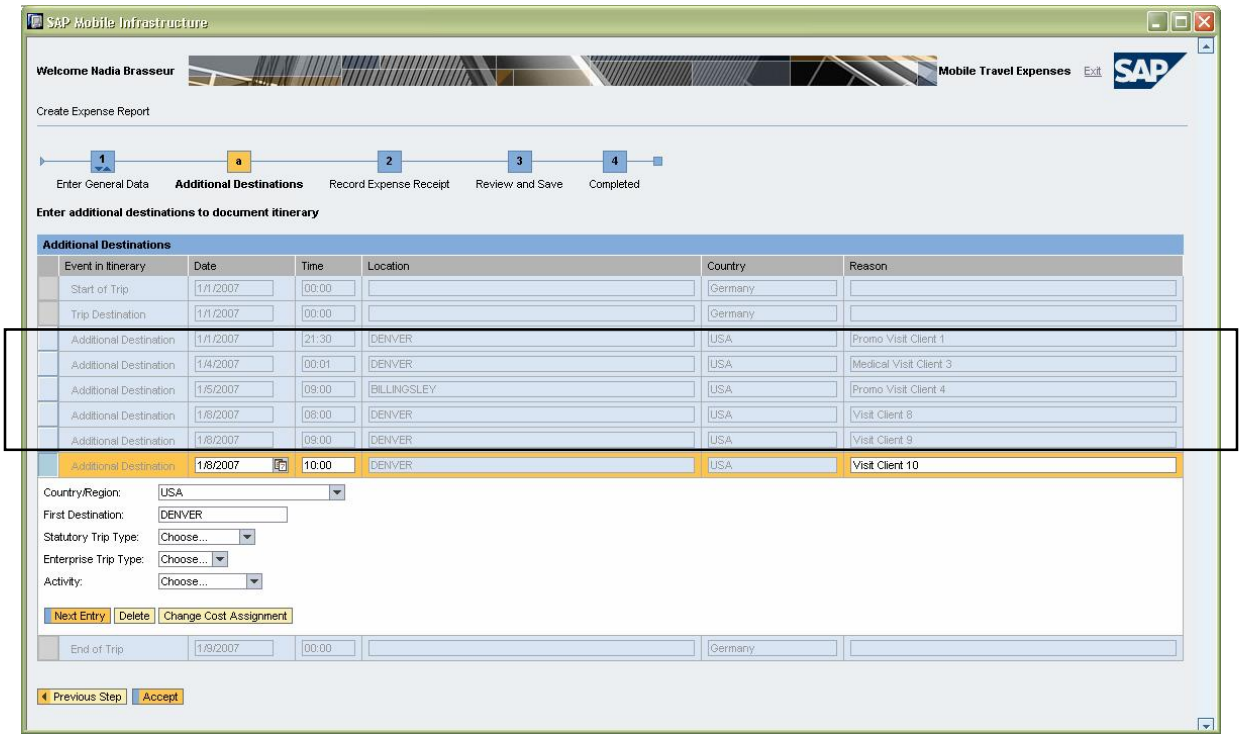
[1] For further information on mCAF see https://www.sdn.sap.com/irj/sdn/weblogs?blog=/pub/wlg/5254

## 2.3 Integration Process Flow

- **Maintain activities**
  You maintain business activities, for example, regarding a customer call or customer site visit, in Mobile Sales for Laptop.
- **Launch Mobile Infrastructure where Mobile Travel Expenses is deployed**
  You launch the Mobile Travel Expenses in the Time and Travel tile set in CRM Mobile. When you do so, the system may prompt you to logon to the Mobile Travel Expenses application. The logon data you enter here is also valid for SAP R/3, meaning you can access both applications with a single sign-on. The user interface for Mobile Travel Expenses now appears. Here, you enter your business trip costs under Trip Costs.
- **Create/Update Trip**
  You enter/update the trip date and time range and any required information for your business trip.
- **System reads activities from SAP CRM Mobile Sales**
  When you enter/update the trip date and time range, the system fetches the business activity data from SAP CRM Mobile Sales to Mobile Travel Expenses. At this point, the system determines all business activities that have business partner information which match the dates you have entered for your business trip. This means several business activities can be determined for one business trip.
- **Pre-populate Additional Destinations**
  System fills business activity data into Additional Destinations. The Additional Destinations page contains the business activity data for those business activities which match the date and time range for the specified business trip.
- **Save trip**
  When you have completed your entries, you save the trip.
- **Synchronize data**
  When you choose Synchronize, the business travel data is synchronized with SAP R/3.

## 2.4 Enhanced Mobile Travel Version

This is what the Additional Page in Mobile Travel Expenses will look like if there are Mobile Sales Activities with Business Partner information that fall within the trip data and time range:




**Note:** Mobile Sales for Laptop uses the Time Zone settings from the Windows settings and Mobile Travel Expenses uses the Time Zone settings from the Mobile Infrastructure settings. To have the Mobile Sales for Laptop Activity times aligned with Mobile Travel Expenses Additional Destinations time, make sure that the Windows and the Mobile Infrastructure have the same Time Zone settings.

# 3 Step-by-Step Solution

The Mobile Travel Expenses Enhancement part requires two components:

- CE_PROPERTIES Add-On
- Mobile Travel Expenses Add-On

To implement the integration scenario as is, please refer to the following note for details and up-to-date information:

- 913352

## 3.1 CE_PROPERTIES Add-On

The CE_PROPERIES Add-On is a .zip file called `CE_POPERTIES.zip` and contains integration properties required for the integration.

This file needs to be assigned and deployed to users as a mobile add-on using the Mobile Travel Expenses deployment mechanism from the Mobile Infrastructure (Web Console or Web Administrator).

This file contains two xml files:

- `CustomerExitProperties.xml`

- `CustomerExitDef.xml`

These two files can be modified by altering the `CE_PROPERTIES.zip` before uploading it to the Mobile Infrastructure server for deployment to all users.

For specific user information, like the login to the Mobile Infrastructure, this requires each user to modify the file after deployment in the folder <*Mobile Infrastructure Installation Directory*>\addons\CE_PROPERTIES. As of SAP CRM Mobile Sales 2007 SP02, users can maintain these settings in the *Time & Travel -> Integration Settings* tile set of MSA.

### 3.1.1 Configuring CustomerExitProperties.XML

The `CustomerExitProperties.XML` stores information about the user and Mobile Sales for Laptop database.

The Mobile Sales for Laptop application takes care of writing credentials and other information to this file, which is then used by the Mobile Travel Expenses application.

Mobile Travel Expenses application reads information from this file for use in the SQL queries.

Here is a sample of the file:

```
<?xml version="1.0"?>
<!DOCTYPE wpf SYSTEM "CustomerExitProperties.DTD">
<wpf>
      <skriptum titel="Properties">
            <SSO>0</SSO>
            <OWNER>73729F5E0B9AD411917408000627B381</OWNER>
            <OWNER_NAME>mobileuser</OWNER_NAME>
            <DSN>ides</DSN>
            <LGN>ides</LGN>
            <PWD>ides</PWD>
            <LOGIN_ME></LOGIN_ME>
            <PWD_ME></PWD_ME>
      </skriptum>
</wpf>
```

#### 3.1.1.1 Owner Properties

For Mobile Travel Expenses to be able to fetch the user's data in the enhancement code using Java SQL APIs, the owner properties need to be accessible.

- OWNER
  Should be left blank. The Mobile Sales for Laptop application writes the user's GUID to this tag when the user logs in to the Mobile Sales for Laptop application.

- OWNER_NAME
  Should be left blank. The Mobile Sales for Laptop application writes the user's ID to this tag when the user logs in to the Mobile Sales for Laptop application.

#### 3.1.1.2 DSN Properties

For Mobile Travel Expenses to be able to fetch the data in the enhancement code using Java SQL APIs, the DSN properties need to be accessible.

- DSN
  DSN being used by MSA
  Should be left blank. The Mobile Sales for Laptop application writes this value to the tag after the user logs in to the Mobile Sales application.

- LGN
  Login being used by the DSN
  Should be left blank. The Mobile Sales for Laptop writes this value to the tag after the user logs in to the Mobile Sales application.

- PWD
  Password being used by the DSN
  Should be left blank. The Mobile Sales for Laptop writes this value to the tag after the user logs in to the Mobile Sales application.

#### 3.1.1.3 Single-Sign-On Mechanism

If you want to avoid having users re-enter their username and password when launching the Mobile Infrastructure and you have not implemented standard SAP SSO (see SAP Guide "How to configure Single Sign-On support of SAP Mobile Infrastructure") functionality, it is possible to implement a mechanism that stores the MTT password in the XML-file. As of MSA 2007 SP02, SAP provides a tool within the Mobile Sales application that allows the encryption of the password (see below – Security Enhancement for MSA and MTT). We recommend the use of the more secure standard SAP SSO.

The Mobile Sales for Laptop application accesses single-sign-on information stored in the `CustomerExitProperties.XML` and behaves accordingly.

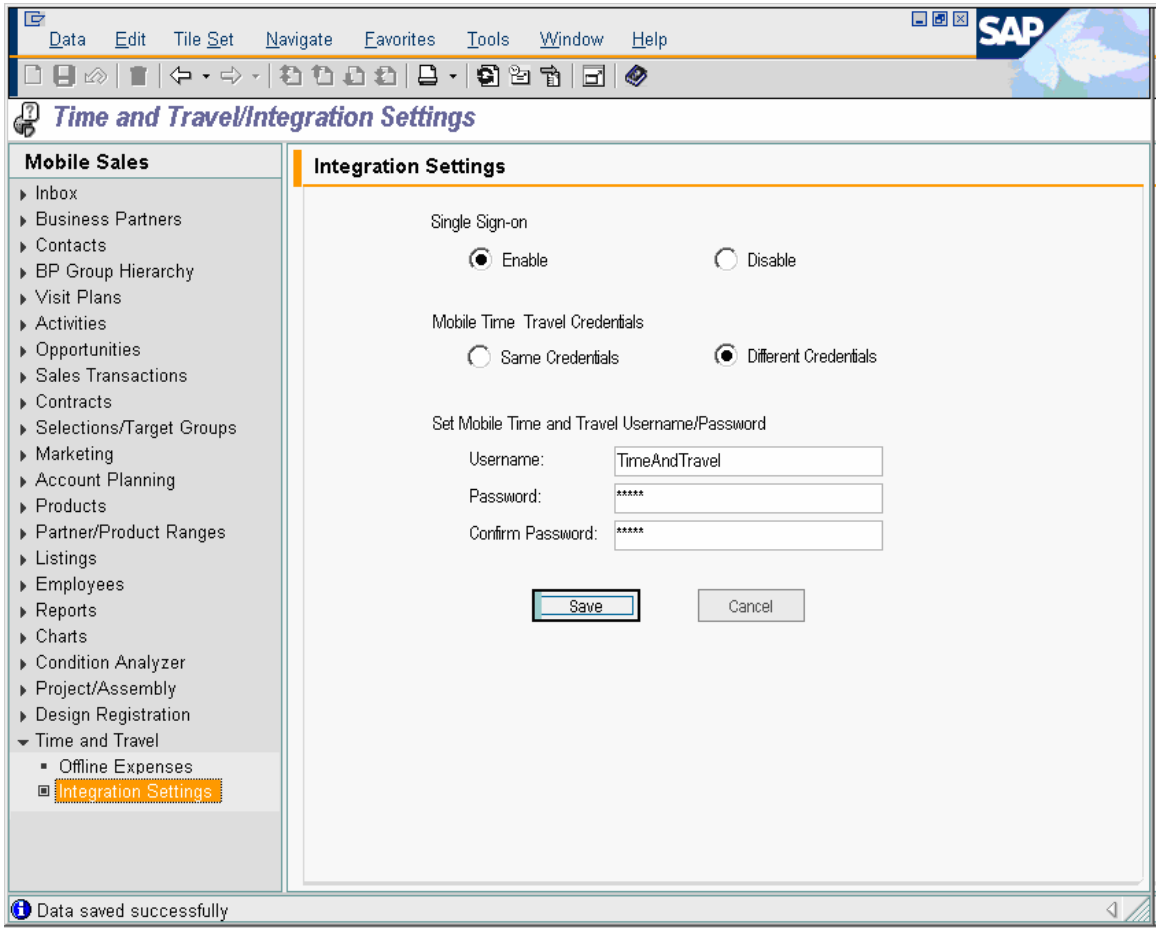
##### 3.1.1.3.1 Security Enhancement for MSA and MTT

MSA's integration with MTT has been enhanced in MSA 2007 SP02 and has improved security features as compared to the earlier versions. A new feature within MSA enables the user to specify integration settings. The user can do the following:

- Enable/disable SSO
- Specify the password required by MTT for automatic login to the MTT application.

The DSN information is read from the registry, encrypted and stored in the CustomerExitProperties.xml file and is used by MTT to access the MSA database. The user's login password for MTT is also encrypted before storing in the database.

Security Enhancement is delivered as of MSA 2007 SP02.




### 3.1.1.3.2 Tool for specifying the integration settings

A tool has been developed in MSA 2007 SP02 that provides a convenient user interface for the procedures that are explained in the following two sections (*Same Username and Password*, *Different Username and/or Password*). The tool is incoporated in MSA as the *Integration Settings* tile set of the Time & Travel component.

MSA uses MSACrypt.dll for encryption which is stored in the path <mobilerootdir>/Bin .NET

MTT also needs to be enhanced for the security enhancement to work. MTT uses MSACrypt.jar to decrypt the password stored in CustomerExitProperties.xml. This jar is also stored in the path <mobilerootdir>/Bin .NET.

CustomerExitProperties.xml after encryption:

```
<?xml version="1.0"?>
<!DOCTYPE wpf SYSTEM "CustomerExitProperties.DTD"[]>
<wpf>
  <skriptum titel="Properties">
    <SSO>0</SSO>
    <OWNER>468CD5976BB53FFBE10000000A421474</OWNER>
    <OWNER_NAME>Mobileuser</OWNER_NAME>
    <DSN>Ides</DSN>
    <LGN>Mobileuser</LGN>
    <PWD>v7dYoZJjOItBuZYpHTslUg==</PWD>
    <LOGIN_ME>TimeAndTravel</LOGIN_ME>
    <PWD_ME>PhFcoN6tqSI0uljo4u2Ung==</PWD_ME>
  </skriptum>
</wpf>
```

The following are code changes required in MTT for the security enhancement to work:

 This security enhancement is valid as of SAP CRM Mobile Sales 2007 SP02 and Mobile Time & Travel 2.0.

1) Make sure that MSACrpyt.jar is in path <mobilerootdir>\Bin.NET

2) Make sure this path is present in listOfJars.txt in the SAP Mobile Infrastructure folder. In any case, MSA writes this path to the file whenever the tool is used.

3) Open MTR 2.0 project in NetWeaver Developer Studio.

4) Right click on the project root and go to Properties > Java Build Path.

5) Add an External Library MSACrypt.jar.

6) Go to the package zcom.sap.mbs.mtr.stop.dataaccess.impl

7) Open the source code of class ZMSADatAccess.java.

8) Import the class: com.sap.msa.MSACrypt;

9) Go to the method loadIdesLoginDetails().

10) Insert the following code below the line: gstrPassword =
nodelist.item(0).getFirstChild().getNodeValue();

```
MSACrypt crypt = new MSACrypt();
gstrPassword = crypt.decrypt(gstrPassword);
```

11) Save, build and test the changes.

12) Export the project to the file MTR.war using the Mobile Development Kit.

#### 3.1.1.3.3 Same Username and Password

If the user has the same username and password for both Mobile Sales for Laptop and Mobile Travel Expenses then you set the <SSO> tag to 1.

When <SSO> tag is set to 1, the Mobile Sales for Laptop would write its username and password to the tags <LOGIN_ME> and <PWD_ME> and when the Mobile Infrastructure is launched, it picks up the username and password from these tags to login automatically.

#### 3.1.1.3.4 Different Username and/or Password

If the user has different usernames and passwords for both Mobile Sales for Laptop and Mobile Travel Expenses, then you should set the <SSO> tag to 0 and enter your Mobile Infrastructure username and password in the tags <LOGIN_ME> and <PWD_ME>.

When <SSO> tag is set to 0, the Mobile Sales for Laptop will not write its username and password to the tags <LOGIN_ME> and <PWD_ME>. It will leave the values entered by the user as is and when the Mobile Infrastructure is launched, it picks up the username and password from these tags to login automatically.

NOTE: The steps mentioned in the section 3.1.1.3.3 and 3.1.1.3.4 are automatically taken care of if the tool (the *Integration Settings* tile set in MSA) is used for specifying the settings.

### 3.1.2 Configuring CustomerExitDef.XML

The Activity data that is fetched by MTT from IDES can be modified by changing the <WHERE_CLAUSEL> tags in the file CustomerExitDef.xml in CE_PROPERTIES.zip.

## 3.2 Mobile Travel Expenses Add-On

The Mobile Travel Expenses Add-On is a .zip file called MTR_CE.zip and contains the enhancement logic to fetch the Mobile Sales for Laptop activity data and display it in Mobile Travel Expenses

This file needs to be assigned and deployed to users as a mobile add-on using the Mobile Travel Expenses deployment mechanism from the Mobile Infrastructure (Web Console or Web Administrator).

This component can be modified by altering the MTR_CE.zip before uploading it to the Mobile Infrastructure server for deployment to all users.

The MTR_CE_SRC.zip contains the source code for the Mobile Travel Expenses Add-On. You can merge this code into your existing development project if you have already implemented

enhancements to the Mobile Time and Travel application or to do any customization. For more information, refer to the Mobile Time and Travel Enhancement Guide available at http://service.sap.com/instguides .

### 3.2.1 Creating the Mobile Travel Expenses Enhancement Package

This section will show you how to enhance the Mobile Travel Expense application to create the integration logic.

The enhancement files created in the following steps are the ones used to create the Mobile Travel Expenses add-on component.

You can also create a new Mobile Travel Expenses development project, implement the integration logic, and export your project into a new MTR.war file which would include the integration logic. Refer to the "Exporting the .war File from SAP NetWeaver Developer Studio (NWDS)" chapter of the Mobile Time and Travel 2.0 Enhancement Guide available at http://service.sap.com/instguides .

#### 3.2.1.1 Prerequisites

Set up the Mobile Travel Expenses projects in NWDS as described in the "Development Cycle" chapter of the Mobile Time and Travel 2.0 Enhancement Guide available at http://service.sap.com/instguides .

#### 3.2.1.2 Procedure

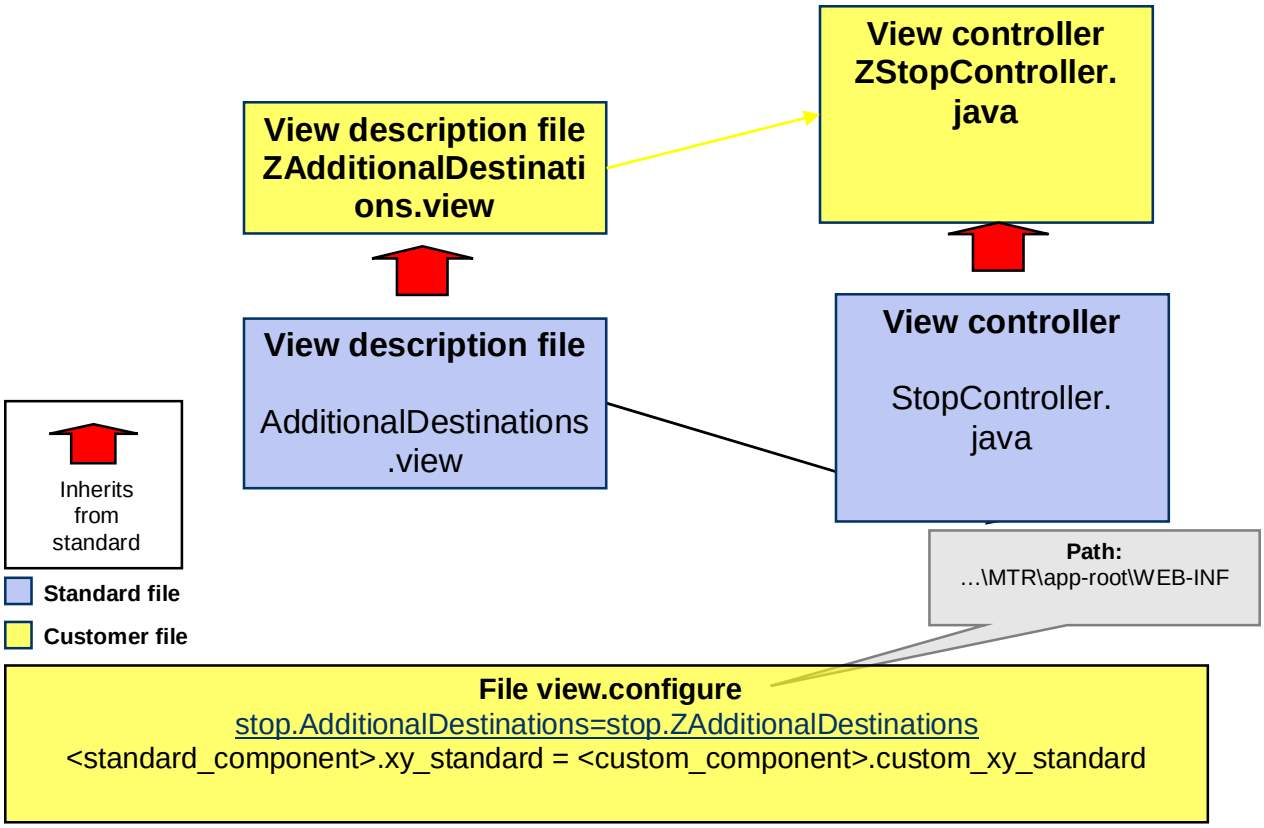
In order to enhance an application, you need to keep the standard coding as is and extend the application to make the application behave in a different way.

The trick to do this is to extend the standard classes and standard view files and configure the application in a different way so that the enhanced classes are used instead of the standard classes.

You extend the standard files by using the mCAF approach. You will extend a view description file and its corresponding view controller. In order for the application to trigger the extended files, you will create a view.configure file. In this file you determine the new view description file to be used.

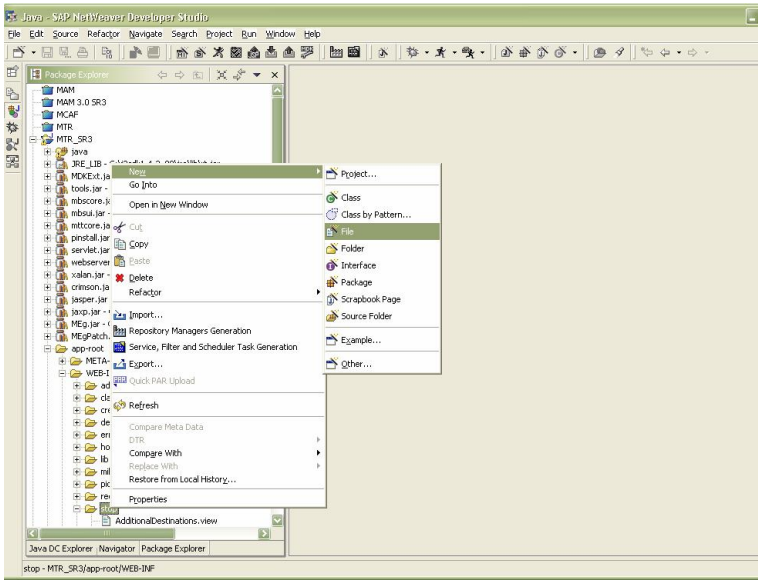
In our case, we will be extending the standard AdditionalDestinations.view and StopController.java files.

You can find more information about this process in the Mobile Time and Travel Enhancement Guide available at http://service.sap.com/instguides .

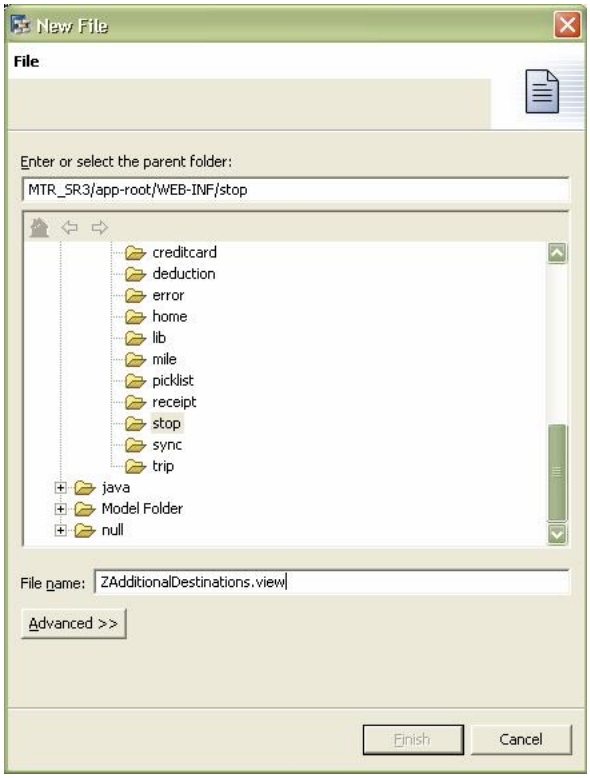

View controller ZStopController. java
View description file ZAdditionalDestinations.view
View description file
AdditionalDestinations .view
View controller
StopController. java
Inherits from standard
Standard file
Customer file
Path:
...\MTR\app-root\WEB-INF
File view.configure
stop.AdditionalDestinations=stop.ZAdditionalDestinations
<standard_component>.xy_standard = <custom_component>.custom_xy_standard

**Step-By-Step Procedure**

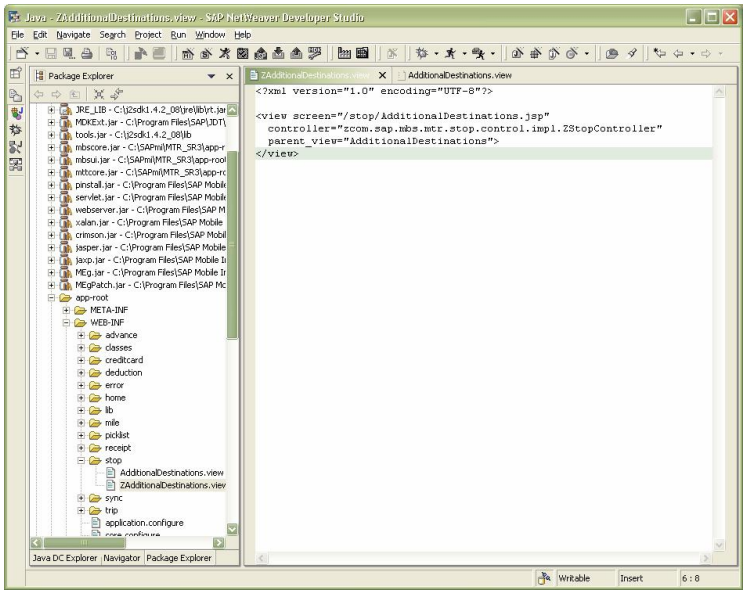
1. Create your new view file. Right-click on `MTR/app-root/WEB-INF/stop`, select “New -> File”



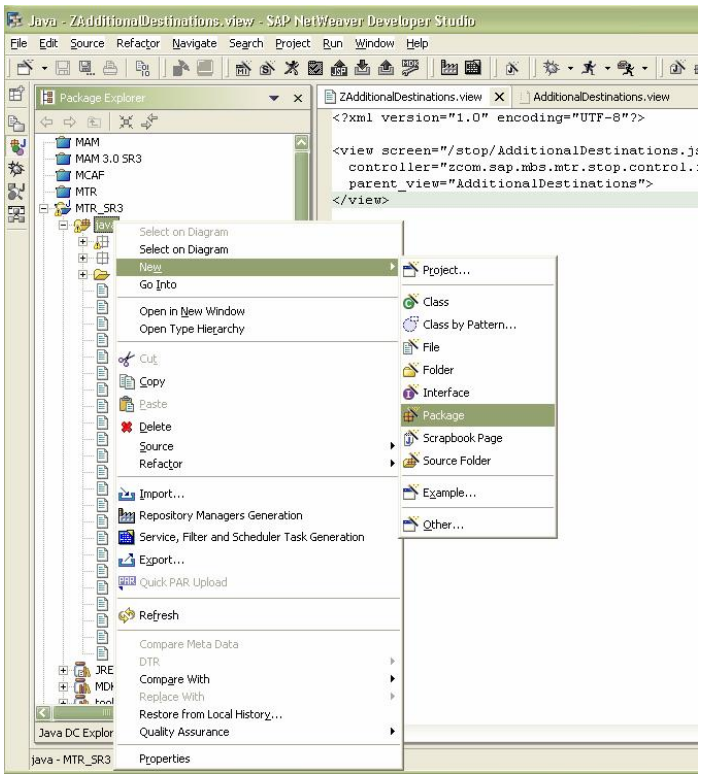

2. Enter “ZAdditionalDestinations.view” and click on Finish



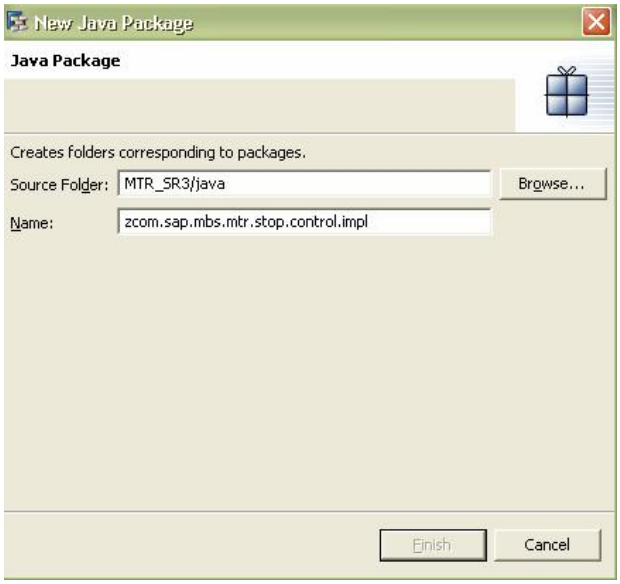
3. Enter the following code which is available in the Appendix:



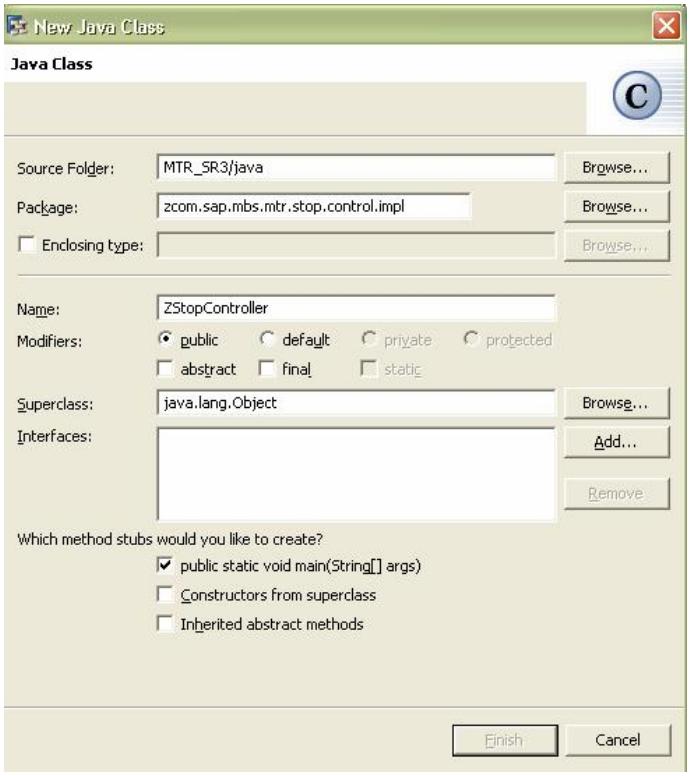

4. Create a new Java package. Right click on `MTR/java`, select “New -> Package”



5. Enter “zcom.sap.mbs.stop.control.impl” and select “Finish”

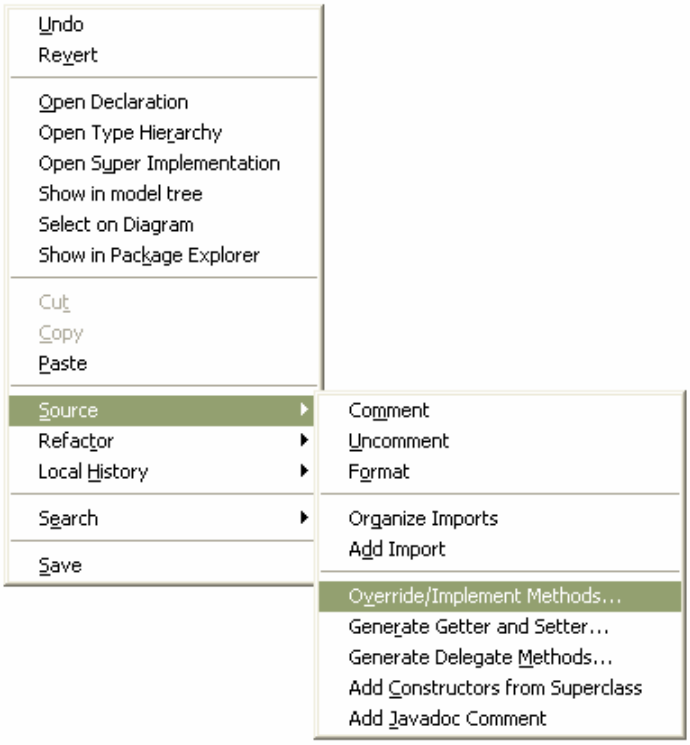



6. Create your new Java Controller. Right-click on the package you created, select “New-> Class” and enter the information as shown here. Click on “Browse” to select the parent Java class.

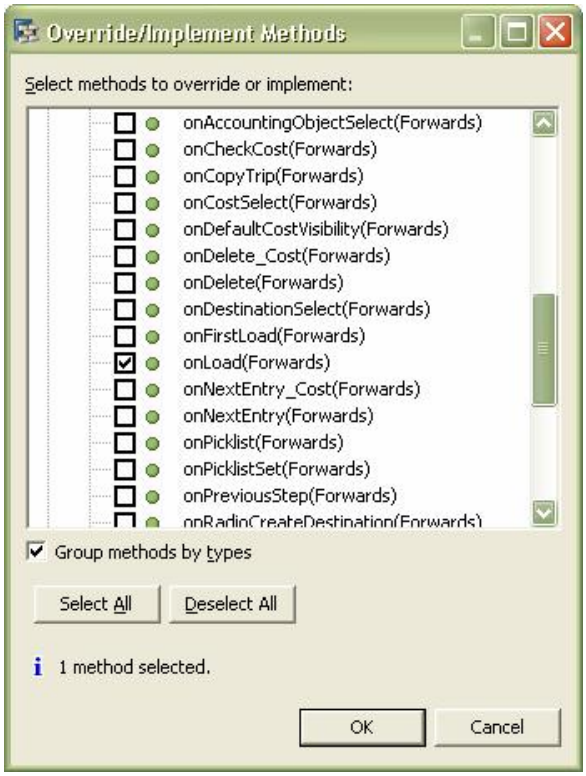


7. Type “StopController”, select the StopController type, and click “OK” then click on Finish.

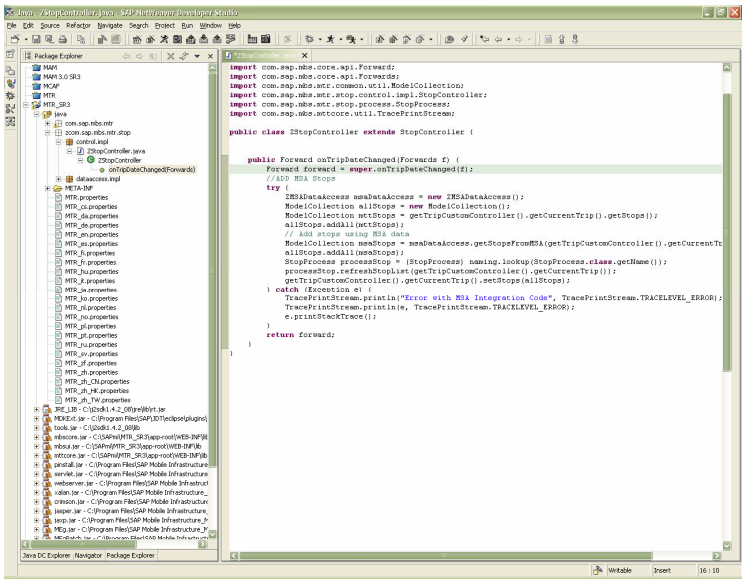

8. In the editor window, right-click and select “Source -> Override/Implement Methods…



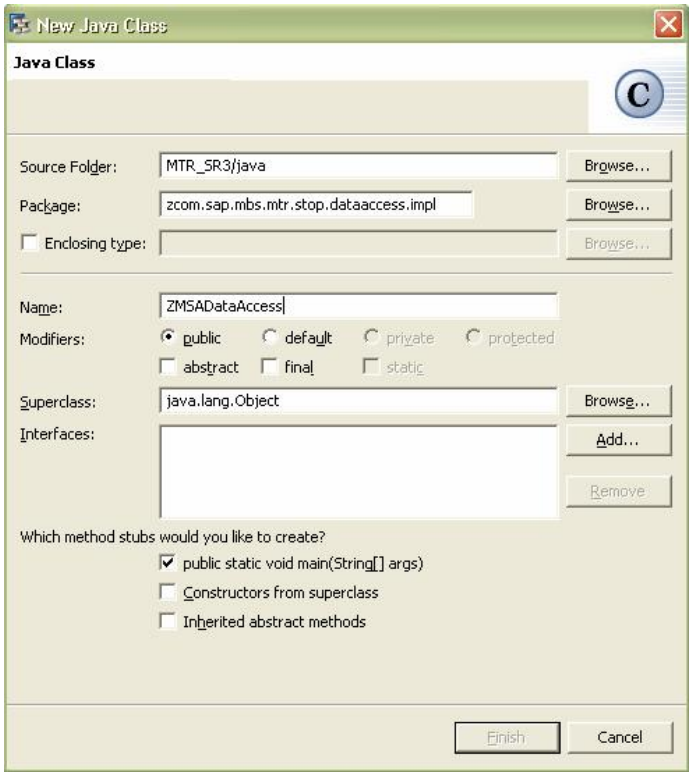
9. Check the onLoad method




10. Enter the following code which is available in the Appendix:

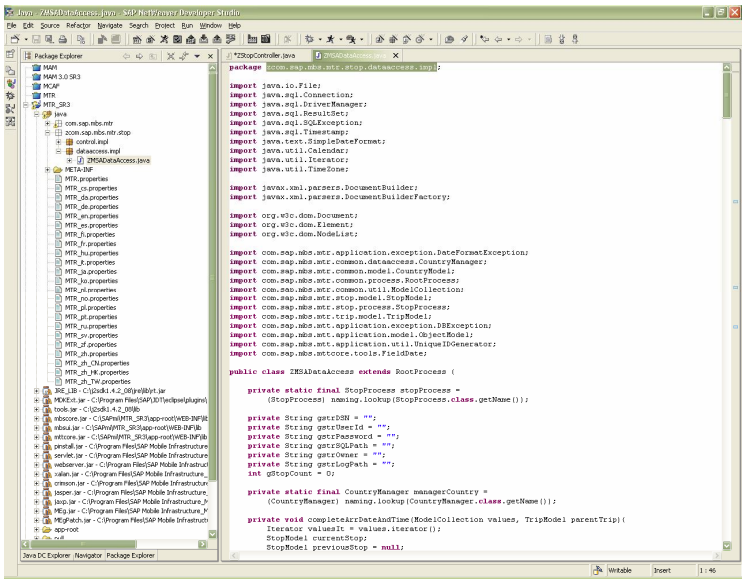


11. Create your new Java data access Class. Right-click on the package you created, select “New -> Class” and enter the information as shown here. Click on “Browse” to select the parent Java class.

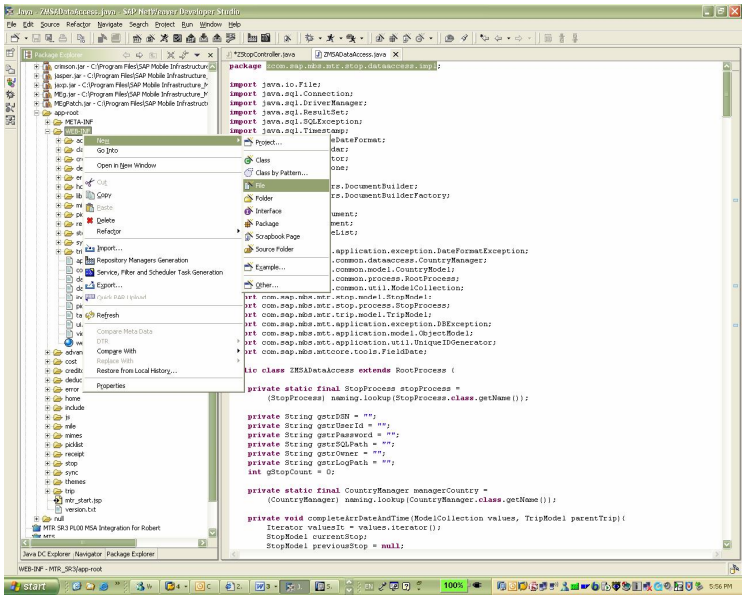



12. Type “RootProcess”, select the RootProcess type, and click “OK” then click on Finish.

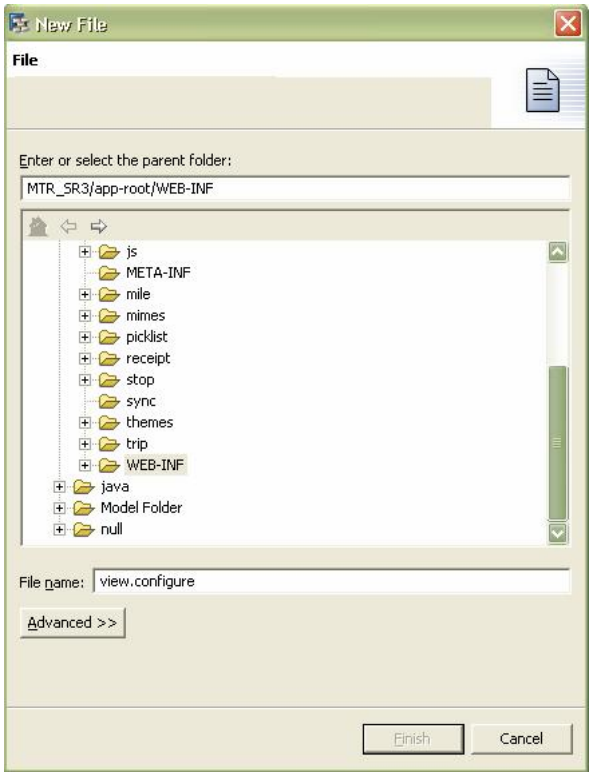
13. Enter the following code which is available in the Appendix:



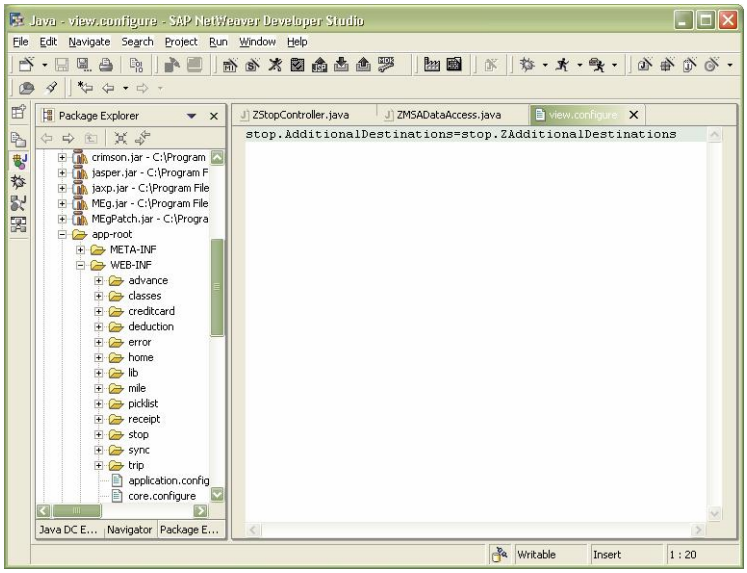
14. Create your `view.configure` file. Right-click on `MTR/app-root/WEB-INF`, select “New -> File”



15. Enter “view.configure” and click on Finish




16. Enter the following code which is available in the Appendix:



17. Save your files, export your project, and test your enhancement logic.

# 4 Appendix

## 4.1 Source Code

Below is the source code for the enhancement files.

You can also find the source code as an attachment (MTR_CE>SRC.zip) in note 913352.

### 4.1.1 ZAdditionalDestinations.view

```
<?xml version="1.0" encoding="UTF-8"?>

<view screen="/stop/AdditionalDestinations.jsp"
  controller="zcom.sap.mbs.mtr.stop.control.impl.ZStopController"
  parent_view="AdditionalDestinations">
</view>
```

### 4.1.2 ZStopController.java

```
 package zcom.sap.mbs.mtr.stop.control.impl;

import zcom.sap.mbs.mtr.stop.dataaccess.impl.ZMSADataAccess;

import com.sap.mbs.core.api.Forward;
import com.sap.mbs.core.api.Forwards;
import com.sap.mbs.mtr.common.util.ModelCollection;
import com.sap.mbs.mtr.stop.control.impl.StopController;
import com.sap.mbs.mtr.stop.process.StopProcess;
import com.sap.mbs.mttcore.util.TracePrintStream;

public class ZStopController extends StopController {



	public Forward onTripDateChanged(Forwards f) {
		Forward forward = super.onTripDateChanged(f);
		//ADD MSA Stops
		try {
			ZMSADataAccess msaDataAccess = new ZMSADataAccess();
			ModelCollection allStops = new ModelCollection();
			ModelCollection mttStops = getTripCustomController().getCurrentTrip().getStops();
```

```
                    allStops.addAll(mttStops);
                    // Add stops using MSA data
                    ModelCollection msaStops =
msaDataAccess.getStopsFromMSA(getTripCustomController().getCurrentTrip(),mttStops);
                    allStops.addAll(msaStops);
                    StopProcess processStop = (StopProcess)
naming.lookup(StopProcess.class.getName());
                    processStop.refreshStopList(getTripCustomController().getCurrentTrip());
                    getTripCustomController().getCurrentTrip().setStops(allStops);
                } catch (Exception e) {
                    TracePrintStream.println("Error with MSA Integration Code",
TracePrintStream.TRACELEVEL_ERROR);
                    TracePrintStream.println(e, TracePrintStream.TRACELEVEL_ERROR);

                    e.printStackTrace();
                }
                return forward;
        }
}
```

### 4.1.3 ZMSADataAccess.java

```
package zcom.sap.mbs.mtr.stop.dataaccess.impl;

import java.io.File;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Timestamp;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Iterator;
import java.util.TimeZone;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;

import com.sap.mbs.mtr.application.exception.DateFormatException;
import com.sap.mbs.mtr.common.dataaccess.CountryManager;
```

```
import com.sap.mbs.mtr.common.model.CountryModel;
import com.sap.mbs.mtr.common.process.RootProcess;
import com.sap.mbs.mtr.common.util.ModelCollection;
import com.sap.mbs.mtr.stop.model.StopModel;
import com.sap.mbs.mtr.stop.process.StopProcess;
import com.sap.mbs.mtr.trip.model.TripModel;
import com.sap.mbs.mtt.application.exception.DBException;
import com.sap.mbs.mtt.application.model.ObjectModel;
import com.sap.mbs.mtt.application.util.UniqueIDGenerator;
import com.sap.mbs.mttcore.tools.FieldDate;

public class ZMSADataAccess extends RootProcess {

	private static final StopProcess stopProcess =
		(StopProcess) naming.lookup(StopProcess.class.getName());

	private String gstrDSN = "";
	private String gstrUserId = "";
	private String gstrPassword = "";
	private String gstrSQLPath = "";
	private String gstrOwner = "";
	private String gstrLogPath = "";
	int gStopCount = 0;

	private static final CountryManager managerCountry =
		(CountryManager) naming.lookup(CountryManager.class.getName());

	private void completeArrDateAndTime(ModelCollection values, TripModel parentTrip){
		Iterator valuesIt = values.iterator();
		StopModel currentStop;
		StopModel previousStop = null;
		while(valuesIt.hasNext()){
			currentStop = (StopModel)valuesIt.next();
			if(previousStop != null){
				previousStop.setARR_DATE(currentStop.getDEP_DATE());
				previousStop.setARR_TIME(currentStop.getDEP_TIME());
			}
			previousStop = currentStop;
		}

		//Need to manually set ARR_DATE and ARR_TIME if no mandatory stops!
		if(!parentTrip.getSchema().allowsMandatoryStops()) {
			if(previousStop != null){
				previousStop.setARR_DATE(parentTrip.getARR_DATE());
```

```
                                        previousStop.setARR_TIME(parentTrip.getARR_TIME());
                                }
                        }
                }

        private String convertToGMT(String dateInLocalFormat, String timeInLocalFormat){
                SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm");
                simpleDateFormat.setTimeZone(TimeZone.getTimeZone("GMT"));
                Calendar calendar =
calendarHelper.parseDateAsCalendar(dateInLocalFormat,timeInLocalFormat);
                String temp = calendar.getTime().toString();
                String result = simpleDateFormat.format(calendar.getTime());
                return result;
        }


        private String convertFromGMT(Timestamp date, Timestamp time) {
                SimpleDateFormat dateFormat = (SimpleDateFormat) FieldDate.getDateFormat();
                String pattern = dateFormat.toPattern();
                SimpleDateFormat simpleDateFormat = new SimpleDateFormat(pattern + " HH:mm");
                        simpleDateFormat.setTimeZone(TimeZone.getDefault());

                //Converting the date (yyyy-MM-dd) to the format yyyyMMdd before calling
parseDateAsLocale
                String convertedDate = date.toString();
                convertedDate =
                                        convertedDate.substring(0,4)
                                        + convertedDate.substring(5,7)
                                        + convertedDate.substring(8,10);

                //Converting the date to the local format before calling parseDateAsCalendar

                String localeFormatDate = calendarHelper.parseDateAsLocale(convertedDate);
                Calendar calendar =
calendarHelper.parseDateAsCalendar(localeFormatDate,time.toString().substring(11,16));
                calendar.setTimeZone(TimeZone.getTimeZone("GMT"));
                String temp = calendar.getTime().toString();
                String result = simpleDateFormat.format(calendar.getTime());
                return result;
        }

        /**
         * Retreives MSA data based on a given trip.
         * @param mTrip TripModel representing the trip for which MSA data should be retrieved.
         * @return ModelCollection containing MSA stop entries for the given trip.
```

```
         * @throws DBException If a new default stop cannot be created.
         * @throws DateFormatException If the dates cannot be retrieved in correct format.
         */
        public ModelCollection getStopsFromMSA(TripModel mTrip, ModelCollection mtrStops)
                throws DBException, DateFormatException {
                ModelCollection colStops = new ModelCollection();
                createConnection();

                //Get the SQL Statement
                String strActQuery = getStatmentFromTag(gstrSQLPath, "skriptum1");
                System.out.println(strActQuery);

                //Convert MTT dates to GMT since GMT used in MSA DB
                String strDepDate = "'" + convertToGMT(mTrip.getDEP_DATE(),
mTrip.getDEP_TIME()).substring(0,10) + "'";
                String strArrDate = "'" + convertToGMT(mTrip.getARR_DATE(),
mTrip.getARR_TIME()).substring(0,10) + "'";
                System.out.println("Dep Date and Arr. Date in GMT : "+strDepDate+" , "+strArrDate);



                //Replace the *Var1* and *Var2* statements defined in the CustomerExitDef.XML file with
actual Trip Dates
                strActQuery = strActQuery.replaceAll("\\*Var1\\*", strDepDate);
                strActQuery = strActQuery.replaceAll("\\*Var2\\*", strArrDate);

                //Add the owner to the where clause
                strActQuery = strActQuery.replaceAll("\\*Var3\\*",gstrOwner);
                System.out.println(strActQuery);

                //Add the sort columns
                strActQuery = strActQuery + " ORDER BY KTABG  ASC ,KTABT  ASC  OPTION (FAST
100)";
                System.out.println(strActQuery);

                try {
                        ResultSet rsActivities =

        getConnection().createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_RE
AD_ONLY).executeQuery(strActQuery);
                                //System.out.println(strActQuery);

                        for (int iRow = 0; rsActivities.next(); iRow++) {
                                //Get the activity and the Customer guids
                                String strActivityGuid = rsActivities.getString("SFAVBKA");
                                String strCustomerGuid = rsActivities.getString("SFAKNA1");
```

```
                                    //TODO: What should be done if the customer guid is not present?
                                    if ((strCustomerGuid == null) || strCustomerGuid.equals("")) {
                                            //do something
                                            continue;
                                    }

                                    //Create the stop and set the arrival and departure
                                    StopModel mStop = stopProcess.getNewDefaultStop(mTrip);
                                    String strStopGuid = UniqueIDGenerator.createID();
                                    mStop.setGuid(strStopGuid);
                                    mStop.setTripGuid(mTrip.getGuid());

                                    //Fetch the arrival and departure date and time
                                    Timestamp dateDeparture = rsActivities.getTimestamp("KTABG");
                                    Timestamp timeDeparture = rsActivities.getTimestamp("KTABT");
                                    Timestamp dateArrival   = rsActivities.getTimestamp("KTAEN");
                                    Timestamp timeArrival   = rsActivities.getTimestamp("KTAET");

                                    //Skip this activity if the start and end time of an activity is same
                                    if(dateDeparture.equals(dateArrival) &&
timeDeparture.equals(timeArrival))
                                    {
                                            continue;
                                    }

                                    //Convert the Departure Date and Time in GMT to the Local TimeZone
before setting it to mStop
                                    String dateTimeInLocal = convertFromGMT(dateDeparture,
timeDeparture);
                                    int pos = dateTimeInLocal.indexOf(" ");
                                    String dateInLocal = dateTimeInLocal.substring(0,pos);
                                    String timeInLocal = dateTimeInLocal.substring(pos+1,pos+6);
                                    mStop.setDEP_DATE(dateInLocal);
                                    mStop.setDEP_TIME(timeInLocal);
                                    System.out.println("Departure Date & Time : "+mStop.getDEP_DATE()+"
"+mStop.getDEP_TIME());

                                    //boolean for checks for duplicate stops in sub-loops
                                    boolean match = false;

                                    //Check if date and time from MSA conflict with previous item from MSA
                                    rsActivities.previous();
                                    try {
                                            String custGuidPrevious = rsActivities.getString("SFAKNA1");
```

```
                                        Timestamp dateDeparturePrevious =
rsActivities.getTimestamp("KTABG");
                                        Timestamp timeDeparturePrevious =
rsActivities.getTimestamp("KTABT");
                                        String ktext = rsActivities.getString("KTEXT");
                                        if ( dateDeparture.compareTo(dateDeparturePrevious)==0 &&
                                                timeDeparture.compareTo(timeDeparturePrevious)==0
&&
                                                !custGuidPrevious.equals("")
                                          ) {
                                                match = true;
                                        }
                                } catch (Exception e){
                                        //Do nothing
                                }
                                //Reset cursor to correct item in ResultSet
                                rsActivities.next();
                                //Continue to next activity if match found since cannot have duplicate stops
                                if (match) {
                                        continue;
                                }

                                //Check if out of range
                                // Stops dep date/time need to be within the trip dep date/time and arr
date/time
                                Calendar dateStop =
calendarHelper.parseDateAsCalendar(mStop.getDEP_DATE());
                                Calendar timeStop =
calendarHelper.parseDateAsCalendar(mStop.getDEP_DATE(),mStop.getDEP_TIME());
                                Calendar depDateTrip =
calendarHelper.parseDateAsCalendar(mTrip.getDEP_DATE());
                                Calendar depTimeTrip =
calendarHelper.parseDateAsCalendar(mTrip.getDEP_DATE(), mTrip.getDEP_TIME());
                                Calendar arrDateTrip =
calendarHelper.parseDateAsCalendar(mTrip.getARR_DATE());
                                Calendar arrTimeTrip =
calendarHelper.parseDateAsCalendar(mTrip.getARR_DATE(), mTrip.getARR_TIME());
                                if (    (dateStop.before(depDateTrip))
                                                ||      // Stop date before trip dep date
                                                (dateStop.equals(depDateTrip) &&
timeStop.equals(depTimeTrip))           ||      // Stop date/time equal to trip dep date/time
                                                (dateStop.equals(arrDateTrip) &&
timeStop.equals(arrTimeTrip))           ||      // Stop date/time equal to trip arr date/time
                                                (dateStop.equals(depDateTrip) &&
timeStop.before(depTimeTrip))           ||      // Stop date equal to trip dep date && stop time before
trip dep time
```

```
                                        (dateStop.equals(arrDateTrip) &&
timeStop.after(arrTimeTrip))           ||   // Stop date equal to trip arr date && stop time after trip
arr time
                                        (dateStop.after(arrDateTrip))
                                                    // Stop date after trip arr date
                                        ){
                                        //Continue to next activity if out of range found since cannot have
out of range stops
                                        continue;
                                }

                                //Check if date and time from MSA conflict with existing stop from MTR
                                for (int i=0; i<mtrStops.size(); i++) {
                                        StopModel stop = (StopModel)mtrStops.get(i);
                                        Calendar dateStopExisting =
calendarHelper.parseDateAsCalendar(stop.getDEP_DATE());
                                        Calendar timeStopExisting =
calendarHelper.parseDateAsCalendar(stop.getDEP_DATE(),stop.getDEP_TIME());
                                        if (dateStopExisting.equals(dateStop) &&
timeStopExisting.equals(timeStop)) {
                                                match = true;
                                                break;
                                        }
                                }
                                //Continue to next activity if match found since cannot have duplicate stops
                                if (match) {
                                        continue;
                                }

                                //Fetch the reason
                                mStop.setCUSTOMER(rsActivities.getString("KTEXT"));

                                //Get the contact person for the activty
                                String strConQuery =
                                        getStatmentFromTag(gstrSQLPath, "skriptum2");
                                //TODO This where clause is dependent on Node order. Replace it with
string replacement
                                strConQuery = strConQuery + "'" + strActivityGuid + "'";

                                ResultSet rsContact =
                                        getConnection().createStatement().executeQuery(strConQuery);
                                System.out.println(strConQuery);

                                //Check if contact person for the activity is present
                                if (rsContact.next() == false) {
                                                //No contact person found. Get Address 1 of the Customer
```

```
                String strCustAddr1Query =
                        getStatmentFromTag(gstrSQLPath, "skriptum3");
                strCustAddr1Query =
                        strCustAddr1Query + """ + strCustomerGuid + """;
                ResultSet rsCustAddr1 =
                        getConnection().createStatement().executeQuery(
                                strCustAddr1Query);
                System.out.println(strCustAddr1Query);

                if (rsCustAddr1.next() == true) {
                        //Found the address 1 of the customer
                        mStop.setLOCATION(rsCustAddr1.getString("CITY1"));
                        String country = rsCustAddr1.getString("COUNTRY");
                        String region = rsCustAddr1.getString("REGION");
                        ObjectModel m =
                                managerCountry.findCountry(
                                        country,
                                        region,
                                        mTrip.getUser());
                        if (m.hasContent()) {
                                mStop.setCountry((CountryModel) m);
                        } else {
                                m =
                                        managerCountry.findCountry(
                                                country,
                                                "",
                                                mTrip.getUser());
                                if (m.hasContent()) {
                                        mStop.setCountry((CountryModel) m);
                                }
                        }

                        gStopCount = gStopCount + 1;
                        mStop.setKEY_STOP(String.valueOf(gStopCount));
                        colStops.add(mStop);
                } else {
                        //TODO : What to be done in this case?
                }
                rsCustAddr1.close();
        } else {
                //Contact person found. Get the Contact person GUID
                String strConGuid = rsContact.getString("SFAKNA1");

                //Get Address 1 of the Contact Person
```

```
                                        String strConAddr1Query =
                                                getStatmentFromTag(gstrSQLPath, "skriptum3");
                                        strConAddr1Query =
                                                strConAddr1Query + """ + strCustomerGuid + """;
                                        ResultSet rsConAddr1 =
                                                getConnection().createStatement().executeQuery(
                                                        strConAddr1Query);
                                        System.out.println(strConAddr1Query);

                                        if (rsConAddr1.next() == true) {
                                                //Found the address 1 of the contact person
                                                mStop.setLOCATION(rsConAddr1.getString("CITY1"));
                                                String country = rsConAddr1.getString("COUNTRY");
                                                String region = rsConAddr1.getString("REGION");
                                                ObjectModel m =
                                                        managerCountry.findCountry(
                                                                country,
                                                                region,
                                                                mTrip.getUser());
                                                if (m.hasContent()) {
                                                        mStop.setCountry((CountryModel) m);
                                                } else {
                                                        m =
                                                                managerCountry.findCountry(
                                                                        country,
                                                                        "",
                                                                        mTrip.getUser());
                                                        if (m.hasContent()) {
                                                                mStop.setCountry((CountryModel) m);
                                                        }
                                                }

                                                gStopCount = gStopCount + 1;
                                                mStop.setKEY_STOP(String.valueOf(gStopCount));
                                                colStops.add(mStop);
                                        }
                                        //No record found for the Contact Person. Display the date,time
and activity description. Location & country will be null
                                        else {
                                                        gStopCount = gStopCount + 1;

        mStop.setKEY_STOP(String.valueOf(gStopCount));
                                                        colStops.add(mStop);
                                        }
                                        rsConAddr1.close();
```

```
                        }
                        rsContact.close();
                }
                rsActivities.close();
        } catch (SQLException se) {
                se.printStackTrace();
        }
        //Set arrival dates and times
        completeArrDateAndTime(colStops, mTrip);
        return colStops;
}

/**
 * Fetches the SQL statement from CustomerExitDef.XML, based on the node name
 * @param strFilePath       Path of the xml file
 * @param strQueryNode  Name of the node containing the XML
 * @return
 */
public String getStatmentFromTag(String strFilePath, String strQueryNode) {
        String strStatement = "";
        try {
                DocumentBuilderFactory documentbuilderfactory =
                        DocumentBuilderFactory.newInstance();
                DocumentBuilder documentbuilder =
                        documentbuilderfactory.newDocumentBuilder();
                Document document = documentbuilder.parse(strFilePath);

                //Fetch the query node
                NodeList nodelist = document.getElementsByTagName(strQueryNode);
                int i = nodelist.getLength();
                for (int j = 0; j < i; j++) {
                        Element element = (Element) nodelist.item(j);
                        NodeList nlFields = element.getElementsByTagName("FELD");
                        int iFieldCount = nlFields.getLength();

                        if (iFieldCount > 0)
                                strStatement = "Select ";
                        //append the select columns
                        for (int l = 0; l < iFieldCount; l++) {
                                Element element1 = (Element) nlFields.item(l);
                                NodeList nlFieldNames =
                                        element1.getElementsByTagName("FELDNAME");
                                NodeList nodelist4 =
                                        element1.getElementsByTagName("ASSIGNED");
```

```
                                        NodeList nodeIsDate =
                                                element1.getElementsByTagName("ISDATE");
                                        NodeList nodeIsTime =
                                                element1.getElementsByTagName("ISTIME");

                                        //append the field name to the list of columns to be selected
                                        strStatement =
                                                strStatement
                                                        +
nlFieldNames.item(0).getLastChild().getNodeValue();

                                        if (l < iFieldCount - 1) {
                                                String s3 = strStatement.concat(", ");
                                                strStatement = s3;
                                        }
                                }

                                //append the table names
                                NodeList nlTables = element.getElementsByTagName("TABELLE");
                                int iTableCount = nlTables.getLength();
                                if (iTableCount > 0) {
                                        strStatement = strStatement + " FROM ";
                                }

                                for (int i1 = 0; i1 < iTableCount; i1++) {
                                        String strTableName =
                                                nlTables.item(i1).getFirstChild().getNodeValue();
                                        strStatement = strStatement + strTableName;

                                        if (i1 < iTableCount - 1) {
                                                strStatement = strStatement + ", ";
                                        }
                                }

                                //append the where clauses
                                NodeList nlWhereClauses =
                                        element.getElementsByTagName("WHERE_CLAUSEL");
                                int iWhereCount = nlWhereClauses.getLength();
                                if (iWhereCount > 0) {
                                        strStatement = strStatement + " WHERE ";
                                }

                                for (int j1 = 0; j1 < iWhereCount; j1++) {
                                        String strWhereClause =
                                                nlWhereClauses.item(j1).getFirstChild().getNodeValue();
```

```
                                        strStatement = strStatement + strWhereClause;
                                        if (j1 < iWhereCount - 1) {
                                                strStatement = strStatement + " AND ";
                                        }
                                }

                        }
                } catch (Exception exception) {
                        exception.printStackTrace();
                }

                return strStatement;
        }
        /**
         * Gets a database connection.
         * @return database connection
         */
        private Connection getConnection() {
                try {
                        return DriverManager.getConnection(
                                "jdbc:odbc:" + gstrDSN,
                                gstrUserId,
                                gstrPassword);

                } catch (Exception e) {
                        e.printStackTrace();
                        return null;
                }

        }

        /**
         * Creates a connection to the database and returns the result. The connection object is assigned to a
global
         * variable.
         * @return true if DB connection is created successfully, false otherwise.
         */
        private boolean createConnection() {
                boolean blnResult = false;

                //Fetch the login details from the properties file
                loadIdesLoginDetails();

                try {
                        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

```
                        //gConn = DriverManager.getConnection("jdbc:odbc:" + gstrDSN, gstrUserId,
gstrPassword);
                } catch (Exception e) {
                        e.printStackTrace();
                        return false;
                }
                return true;
        }

        /**
         * Fetches the login details for the IDES database from CustomerExitProperties.xml and
         * stores them in global variables
         */
        private void loadIdesLoginDetails() {
                try {
                        File dirCurrent = new File(System.getProperty("user.dir"));
                        String path = dirCurrent.getPath();
                        dirCurrent = dirCurrent.getParentFile().getParentFile();
                        File fileProperties =
                                new File(
                                        dirCurrent.getAbsolutePath()
                                                +
"\\addons\\CE_PROPERTIES\\CustomerExitProperties.XML");
                        gstrSQLPath =
                                dirCurrent.getAbsolutePath()
                                        + "\\addons\\CE_PROPERTIES\\CustomerExitDef.XML";

                        //Parse the xml
                        DocumentBuilderFactory documentbuilderfactory =
                                DocumentBuilderFactory.newInstance();
                        DocumentBuilder documentbuilder =
                                documentbuilderfactory.newDocumentBuilder();
                        Document document = documentbuilder.parse(fileProperties);

                        NodeList nodelist = document.getElementsByTagName("OWNER");
                        if (nodelist.getLength() > 0) {
                                gstrOwner = nodelist.item(0).getFirstChild().getNodeValue();
                        }

                        nodelist = document.getElementsByTagName("DSN");
                        if (nodelist.getLength() > 0) {
                                gstrDSN = nodelist.item(0).getFirstChild().getNodeValue();
                        }

                        nodelist = document.getElementsByTagName("PWD");
```

```
                        if (nodelist.getLength() > 0) {

                                //PWD tag is null in CustomerExitProperties.xml
                                if (nodelist.item(0).getFirstChild() == null) {
                                        //System.out.println("pwd is null");
                                        gstrPassword = null;
                                } else
                                        gstrPassword =
                                                nodelist.item(0).getFirstChild().getNodeValue();
                        }

                        nodelist = document.getElementsByTagName("LGN");
                        if (nodelist.getLength() > 0) {
                                gstrUserId = nodelist.item(0).getFirstChild().getNodeValue();
                            MSACrypt crypt = new MSACrypt();
                            gstrPassword = crypt.decrypt(gstrPassword);
                        }
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }
}
```

#### 4.1.4 view.configure

```
stop.AdditionalDestinations=stop.ZAdditionalDestinations
```

www.sap.com/mobile