
WEBFLEET.connect

Driver safety and efficiency

Revision history	3
1 Introduction	4
1.1 Example use case	4
1.2 Editorial note: Code samples	5
2 Creating an overview	6
2.1 Creating a simple overview	6
2.2 Extending the overview	8
3 Filtering the results	10
4 Analysing the OptiDrive numbers	12
5 Drilling down into the information	14
6 Retrieving settings for the OptiDrive Indicator	15



Revision history

TomTom WEBFLEET.connect

TomTom Business Solutions, WEBFLEET.connect Driver safety and efficiency, version 1.0.0

Copyright © 2011 TomTom Business Solutions, TomTom International B.V., all rights reserved

No part may be reproduced except as authorised by written permission. The copyright and the foregoing restriction extend to reproduction in all media.

Table: Revision history:

Revision	Date	Description	Author
1.0.0	2011-08-02	Initial release	RH

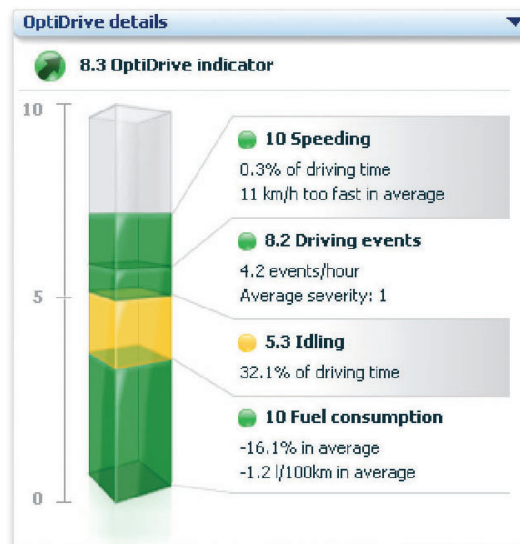
1 Introduction

Using the WEBFLEET OptiDrive Indicator, fleet managers can easily reward safe and efficient driving and at the same time offer training to drivers who have room for improvement.

The OptiDrive Indicator shows the safety and efficiency of the driver's driving style ranging from 0 (needs improvement) to 10 (excellent). It considers idling, speeding, harsh driving and fuel consumption information. The values and relative importance of each of these four variables are used to calculate the OptiDrive Indicator.

After analysing the four variables you could choose to reward or train your drivers according to their needs. That makes the driver feel more appreciated and safer, and the business saves on maintenance and fuel costs and reduce its carbon footprint.

The intention of this article is to give you ideas on how to work with the OptiDrive Indicator and the related information. You will learn how to create a tool to detect the drivers who need to reduce their fuel consumption and the time they are idling or change their harsh and fast driving style. With it, you will also be able to discover which drivers you want to reward for their excellent driving style.



For information about the calculation done by the OptiDrive Indicator, see [Retrieving settings for the OptiDrive Indicator](#).

1.1 Example use case

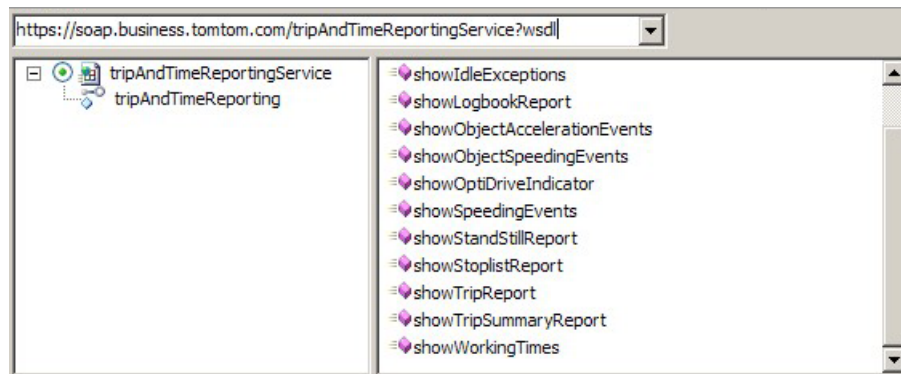
Company XYZ has been given a public contract with preconditions. The first precondition is that the vehicle fleet must be driven safely. The second precondition is that the fleet must be driven in an environmentally responsible way to protect both people and the environment. In addition, XYZ themselves are aiming to reduce their fuel consumption as well as the maintenance cost of their fleet.

To motivate their drivers to improve their driving styles, the company wants to offer incentives. Drivers in the top third for driving style performance shall receive bonuses for their outstanding driving style. The drivers in the lower third shall receive training where they have a need for improvement. Drivers in the middle third already have a solid driving style and will neither receive training nor bonuses.

The company has just upgraded to the WORKsmart-Safety solution from TomTom Business Solutions. Using this solution, the safety and efficiency of the drivers' driving style can be easily evaluated. You have been asked to integrate a tool for the analysis of WEBFLEET data.

1.2 Editorial note: Code samples

This guide shows code samples written in C#. The samples use the TomTom WEBFLEET.connect SOAP interface. To understand the examples, we assume that SOAP endpoint address `tripAndTimeReportingService` has been added to a Visual Studio project:



We offer complete example projects on our website. To download the example projects go to <http://business.tomtom.com/solutions/integration/resources/>. To use the examples you need to have Visual Studio 2008 installed. In the example code look at the class `OptiDriveServiceHandler`.

To show how the results can be obtained, all information is copied from the result objects of the Webservice to a custom object of type `OptiDriveData`. In the example this type is also used to display the results in a table.



2 Creating an overview

2.1 Creating a simple overview

Let's first create a simple overview and include the driver number, driver name and the OptiDrive Indicator to see the overall performance at a glance.

You will carry out the following steps:

1. Indicate all needed request parameters.
2. Make the actual SOAP call.
3. Check the resulting response code.
4. Copy the retrieved data into the self-defined object.
5. Notify event listeners about new data (for example graphical user interface).

Note: *The following example shows the complete method `GetOptiDriveValuesWorker ()`. The rest of the examples in this guide show only code snippets.*

Code sample 2-1: Creating a basic set of OptiDrive information

```
private void GetOptiDriveValuesWorker (string account, string user,
string password)
{
// Fill all needed request parameters
AuthenticationParameters aParam = new AuthenticationParameters();
    aParam.accountName = account; aParam.userName = user;
    aParam.password = password;
GeneralParameters gParam = new GeneralParameters();
    gParam.locale = KnownLocales.DE; gParam.timeZone =
    KnownTimeZones.Europe_Berlin;
DriverScoreParameter optiDriveParam = new DriverScoreParameter();
    optiDriveParam.dateRange = new DateRange();
    optiDriveParam.dateRange.rangePattern = DateRangePattern.WF0;
// make actual SOAP call
tripAndTimeReportingClient tripClient =
    new tripAndTimeReportingClient();
GenericServiceQueryOpResult optiDriveResult =
    tripClient.showOptiDriveIndicator(aParam, gParam, optiDriveParam);
// Check result response code
IList<OptiDriveData> resultList = new List<OptiDriveData>();
    if (optiDriveResult.statusCode == 0)
    {
        foreach (TransferObjectBase obj in optiDriveResult.results)
        {
// Copy data into self defined object 'OptiDriveData'
OptiDriveIndicator optiDriveItem = obj as OptiDriveIndicator;
OptiDriveData optiDriveData = new OptiDriveData();
optiDriveData.DriverNo = optiDriveItem.driver.driverNo;
optiDriveData.DriverName = optiDriveItem.driver.driverName;
optiDriveData.OptiDriveIndicator = optiDriveItem.optiDriveIndicator;
// Add self-defined object to list
resultList.Add(optiDriveData);
        }
    }
    else
    {
// skipped proper error handling in this example
MessageBox.Show(optiDriveResult.statusMessage, "OptiDriveExample");
    }
// now resultList contains OptiDriveData entries
// Notify event listeners about new data (e.g. GUI)
    OnOptiDriveDataAvailable(resultList);
}
}
```

Table 2-1: Sample results:

Driver no.	012	013	014	015
Driver name	Peter P.	Marc M.	Craig C.	Dirk D.
OptiDrive Indicator	0.831	0.612	0.263	0.015

Understanding the results

Use the OptiDrive Indicator to recognise the driving style. The value provided by WEBFLEET.connect ranges from 0 to 1. With it you can assign drivers to the three categories, for example:

- 0.667 to 1.0 = excellent = provide a reward
- 0.334 to 0.666 = good
- 0 to 0.333 = needs improvement = provide a training

2.2 Extending the overview

You can extend the information in the overview by adding the total driving time and the total distance driven. You can also add the trend to show if driving style worsened, improved or was stable over a defined time and distance.

Code sample 2-2: Extending OptiDrive information by additional information about time, distance and trend

```

optiDriveData.TotalTime =
    TimeSpan.FromSeconds(optiDriveItem.totalDrivingTime);
optiDriveData.TotalDistance = optiDriveItem.totalDistance / 1000;
optiDriveData.Trend = optiDriveItem.trend;
    
```

Table 2-2: Sample results:

Driver no.	012	013	014	015
Driver name	Peter P.	Marc M.	Craig C.	Dirk D.
OptiDrive Indicator	0.831	0.612	0.263	0.015
Total time	12:23 h	12:05 h	09:55h	00:01h
Total distance	1020 km	995 km	1122km	0,05km
Trend	0.4	0.1	-0.1	-0.9

Understanding the results

Look at the explanations below together with the trends indicated in the table above to understand the driving results:

- Driver 012 - Excellent driver who has strongly improved his driving style over the last few days.
- Driver 013 - Good driver who has slightly improved his driving style over time.
- Driver 014 - Belongs to the lower third and needs training. His driving style has got slightly worse.
- Driver 015 - Probably had low use of the vehicle. He had possibly moved the vehicle from one parking lot to another, which resulted in a very strong decrease of performance and a bad OptiDrive Indicator. This result may not be representative and should be excluded from the evaluation.

Note: *Time and distance help you understand if the values are representative for each driver, compared to the period you have defined.*

3 Filtering the results

You could filter the results for those drivers belonging to the upper third with an OptiDrive Indicator greater than 6.7 for bonuses and those belonging to the lower third with an OptiDrive Indicator smaller than 3.4 for training.

Note: For the following example the user interface has been extended by a second table. The upper table is now used for the high performing drivers. The lower table is used for drivers with training needs. The filtering is done in the method `FillDataGridView` located in the `MainDialog` class.

Code sample 3-1: Filtering the OptiDrive values for the upper and the lower third

```
private void FillDataGridView(IList<OptiDriveData> data)
{
    // read values from GUI
    double upperBound = (double)upperNumericUpDown.Value;
    double lowerBound = (double)lowerNumericUpDown.Value;

    IList<OptiDriveData> upperList = new List<OptiDriveData>();
    IList<OptiDriveData> lowerList = new List<OptiDriveData>();
    foreach (OptiDriveData optiItem in data)
    {
        if (optiItem.OptiDriveIndicator > upperBound)
        {
            upperList.Add(optiItem);
        }
        else if (optiItem.OptiDriveIndicator < lowerBound)
        {
            lowerList.Add(optiItem);
        }
    }
    this.upperDataGridView.DataSource = upperList;
    this.lowerDataGridView.DataSource = lowerList;
}
```

Table 3-1: Sample results for the upper third:

Driver no.	012
Driver name	Peter P.
OptiDrive Indicator	0.831
Total time	12:23 h
Total distance	1020 km
Trend	0.4

Table 3-2: Sample results for the lower third:

Driver no.	014	015
Driver name	Craig C.	Dirk D.
OptiDrive Indicator	0.263	0.015
Total time	09:55h	00:01h
Total distance	1122km	0,05km
Trend	-0.1	-0.9

4 Analysing the OptiDrive numbers

Now that you have assigned the drivers to the three categories, you can analyse their needs in order to find the right training.

Next, we can retrieve for each drive the four variables in the OptiDrive Indicator. They show in which of the following aspects your drivers are weak or strong for speeding, harsh driving, idling and fuel consumption. Their ranges compare to those of the OptiDrive Indicator.

Note: *The following example takes more values from the Webservice results. It also adds a flag that indicates drivers who have a weak speeding indicator. The flag is shown in the table in the GUI.*

Code sample 4-1: Adding the four variables to the result list

```
optiDriveData.SpeedingIndicator = optiDriveItem.speedingIndicator;
optiDriveData.DrivingEventsIndicator =
    optiDriveItem.drivingEventsIndicator;
optiDriveData.IdlingIndicator = optiDriveItem.idleIndicator;
optiDriveData.FuelusageIndicator = optiDriveItem.fuelUsageIndicator;
// detect fast drivers (using a hardcoded threshold for this example)
if (optiDriveData.SpeedingIndicator < 0.3)
{
    optiDriveData.TooFast = true;
}
resultList.Add(optiDriveData);
```

Table 4-1: Sample results for the upper third:

Driver no.	014
Driver name	Craig C.
OptiDriver Indicator	0.263
Total time	09:55h
Total distance	1122km
Trend	-0.1
Speeding	0.072
Driving events	0.102
Idling	0.361
Fuel consumption	0.515

Understanding the results

The low numbers for speeding and driving events indicate an unsafe driving style. The driving style of driver 014 is not environmentally responsible either if you look at the indicator for idling. The tool you integrate could possibly highlight this information and lead to the right decision for appropriate training for driver 014.

***Note:** Even drivers with a good or excellent driving style, according to the OptiDrive Indicator, may have weaknesses in one of the four skills. It pays to look into the details of those drivers and possibly offer training to them too, to keep the performance of the whole fleet on a highly safe and efficient level.*



5 Drilling down into the information

You can bring more transparency to the previously aggregated numbers by providing additional information. For example for speeding, you could show how often a driver was speeding, the average speed he was going too fast compared to the average speed limit during that time and how long he was speeding in total. The same applies to driving events, idling and fuel consumption.

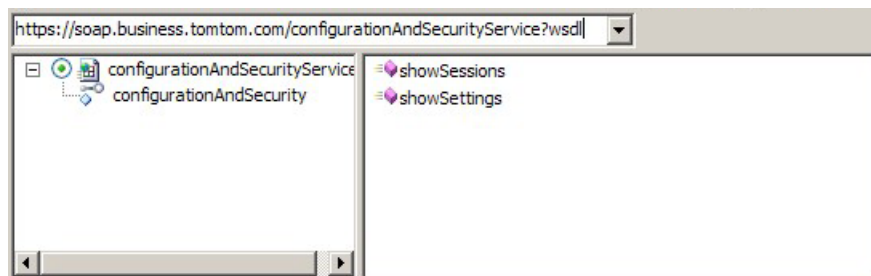
You can drill down into the information as much as you want by using additional functions such as `showSpeedingEvents`, `showAccelerationEvents`, `showIdleExceptions`, and `showVehicleReportExtern` for fuel consumption information per vehicle.

Note: *The OptiDrive Indicator reflects results that have been recorded during a defined period for the specified driver across multiple vehicles. Hence, you can do further analysis by looking at the results for a specific vehicle.*

6 Retrieving settings for the OptiDrive Indicator

To gain more transparency for the calculation of the OptiDrive Indicator you can retrieve the respective settings by using `showSettings`. You can then see which variables are influencing the calculation and the weight given to them. This is defined in the WEBFLEET account settings.

Note: To read the settings add another SOAP endpoint `configurationAndSecurity`:



Code sample 6-1: Retrieve OptiDrive settings

```

private OptiDriveSettings GetOptiSettings(string account,
                                         string user,
                                         string password)
{
// fill request parameters
    ConfigService.AuthenticationParameters aParam
        = new ConfigService.AuthenticationParameters();
    aParam.accountName = account;
    aParam.userName = user;
    aParam.password = password;

    ConfigService.GeneralParameters gParam
        = new ConfigService.GeneralParameters();
    gParam.locale = ConfigService.KnownLocales.DE;
    gParam.timeZone = ConfigService.KnownTimeZones.Europe_Berlin;

// make actual call
    ConfigService.GenericServiceQueryOpResult settingsResult =
        this.configClient.showSettings(aParam, gParam);

    if (settingsResult.statusCode == 0)
    {
        ConfigService.Settings settings
            = settingsResult.results[0] as ConfigService.Settings;
        OptiDriveSettings optiSettings = new OptiDriveSettings();
        optiSettings.Driving = settings.drivingComponentWeightOptiDrive;
        optiSettings.Fuel = settings.fuelUsageComponentWeightOptiDrive;
        optiSettings.Idling = settings.idlingComponentWeightOptiDrive;
        optiSettings.Speeding = settings.speedingComponentWeightOptiDrive;
        return optiSettings;
    }
    return null;
}

```

Table 6-1: The overview of variables influencing the calculation of the OptiDrive Indicator and the weight given to them:

Idling	Speeding	Driving events	Fuel usage
0.253	0.253	0.253	0.241

Understanding the results

The table above shows that all variables are used in the calculation of the OptiDrive Indicator. The results for idling, speeding and driving events are given a higher weight than the results for fuel consumption in the calculation of the OptiDrive Indicator for this WEBFLEET account. This could indicate that the fleet manager

measures the driving style of his drivers more against their driving behaviour than against their fuel consumption.

The four variables are as follows:

- **Fuel consumption**

(ecoPLUS required)

- The average amount of fuel used measured against a reference value in l/100km or mpg.
- The average fuel consumption in relation to the reference value in %.

- **Speeding**

(Working with all TomTom devices)

- How long the driver was breaking the speed limits in relation to the total driving time.
- An average indication of if and by how much the driver drove too fast.

- **Driving events**

(TomTom LINK 300/310 required)

- The number of driving events per hour.

A driving event is reported to WEBFLEET when a driver exceeds a certain level of acceleration. Acceleration is recorded during both braking, cornering and steering.

- The average severity of the recorded driving events ranging from 1 to 5.

- **Idling**

(TomTom LINK/LINK 300/310 connected to IGN)

- How much time the driver let the vehicle stand still with the engine running.
- How much fuel did the driver waste through idling in relation to the total fuel consumption.

Note: You don't necessarily need to use the OptiDrive values provided by TomTom but you can create your own calculation scheme by making use of all information provided by WEBFLEET with regards to speeding events, acceleration events, idling events and fuel consumption, see [Drilling down into the information](#).