SAP ME OEE Customization Guide

Version 1.0

Applicable Release: SAP ME 15.2

Revision History

Date	Version	Description	Author
07/12/2018	1.0	Initial revision	Konstantin Kireyev

© Copyright SAP SE 2018

Contents

1 Summary 4

2 References 4

3 Requirements 5

4 Prerequisites 5

5 Custom Resource Statuses 6

6 Custom OEE KPI Targets 7

7 Custom KPI Formula 8

- 7.1 Activity Maintenance 8
- 7.2 Service Extension Maintenance 8
- 7.3 Service-config.xml 9
- 7.4 Service Extension Class 10
- 7.5 OEE KPI Calculation 11
- 7.6 OEE Time Model Terminology 11
- 7.7 FindKpiParametersResponse 11
- 7.8 Customization Scenarios 13

© Copyright SAP SE 2018

SAP ME OEE Customization

1 Summary

The purpose of this document is to outline the required changes to embed custom OEE logic in SAP ME OEE POD.

2 References

The following documents define the requirements addressed by this design:

Document Type	Document Title
Developer Guide	SAP Manufacturing Execution Customization
Help Page	Overall Equipment Effectiveness

© Copyright SAP SE 2018

3 Requirements

Customers may have a need to do the following:

- Create custom statuses for resources.
- Define custom OEE KPI targets for a resource.
- Embed custom logic into KPI calculation in OEE POD.

4 Prerequisites

- The custom developer's guide describes how to create service extension activities (see *Service Extension* section) using SAP NetWeaver Developer Studio (NWDS).

© Copyright SAP SE 2018

5 Custom Resource Statuses

SAP ME Data Exchange Interface (available from page `\manufacturing-xml`) can be used to create a new custom status for resources. Here is an example of a data exchange request to create a custom status with code “7” and description “CustomStatus” on site “OEE”:

```
<DX_REQUEST>
   <IMPORT mode="both">
            <STATUS_OBJECT>
                  <ROW>
                        <DEFAULT_STATUS>false</DEFAULT_STATUS>
                        <CREATED_DATE_TIME_ISO>2017-11-
02T15:30:19.000Z</CREATED_DATE_TIME_ISO>

   <STATUS_DESCRIPTION>CustomStatus</STATUS_DESCRIPTION>
                        <STATUS>7</STATUS>
                        <MODIFIED_DATE_TIME_ISO>2017-11-
02T15:30:19.000Z</MODIFIED_DATE_TIME_ISO>
                        <STATUS_MEANING>
                              <ROW>

   <STATUS_MEANING>true</STATUS_MEANING>

   <STATUS_MEANING_TYPE>Enabled</STATUS_MEANING_TYPE>
                              </ROW>
                              <ROW>

   <STATUS_MEANING>true</STATUS_MEANING>

   <STATUS_MEANING_TYPE>Utilized</STATUS_MEANING_TYPE>
                              </ROW>
                        </STATUS_MEANING>
                        <STATUS_GROUP>RESOURCE</STATUS_GROUP>
                        <SITE>OEE</SITE>
                  </ROW>
            </STATUS_OBJECT>
   </IMPORT>
</DX_REQUEST>
```

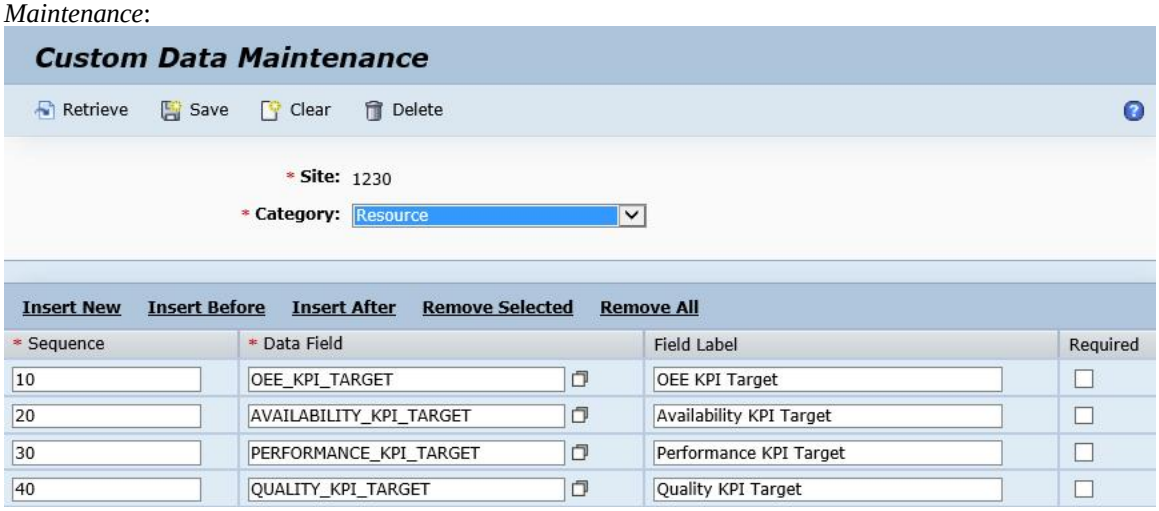
When the custom status is created it must be assigned to a time element type in *Time Element Type Assignment* activity to be utilized in OEE calculations.

© Copyright SAP SE 2018

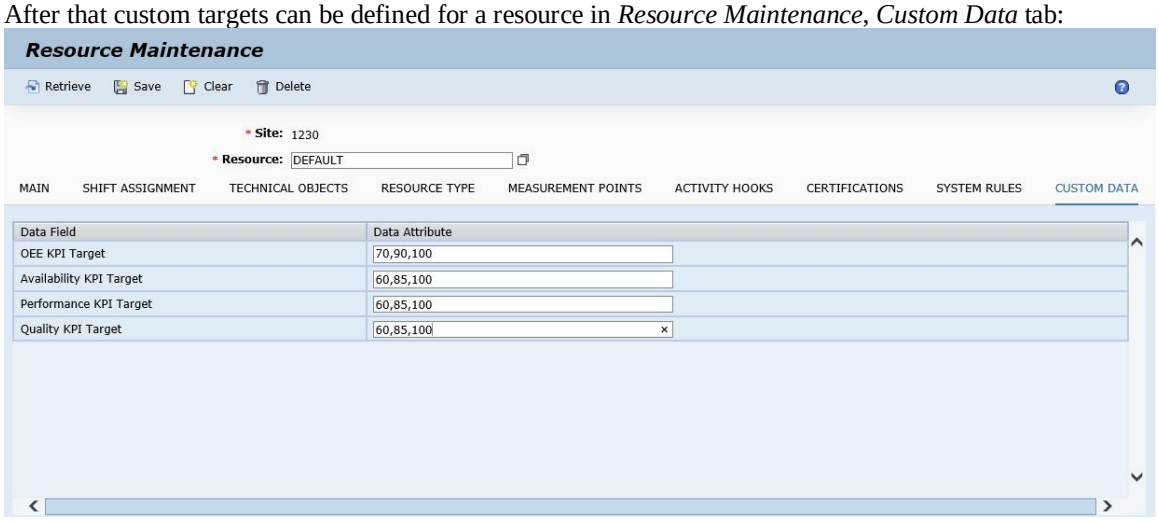
6 Custom OEE KPI Targets

It is possible to define custom KPI targets for a resource.

The following *Custom Data Field* definitions must be defined for Category "Resource" in *Custom Data Maintenance*:



After that custom targets can be defined for a resource in *Resource Maintenance*, *Custom Data* tab:



If custom KPI targets are defined, they will be applied when displaying color code (red, yellow, green) for calculated KPI in OEE POD for the selected resource:



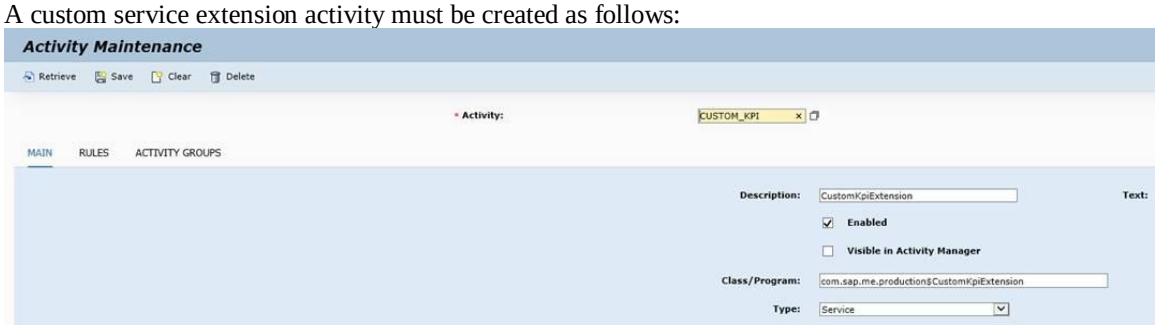
© Copyright SAP SE 2018

7 Custom KPI Formula

The formula to calculate OEE KPIs can be extended by a custom service extension activity (see the custom developer's guide, *Service Extension* section) created in NWDS.

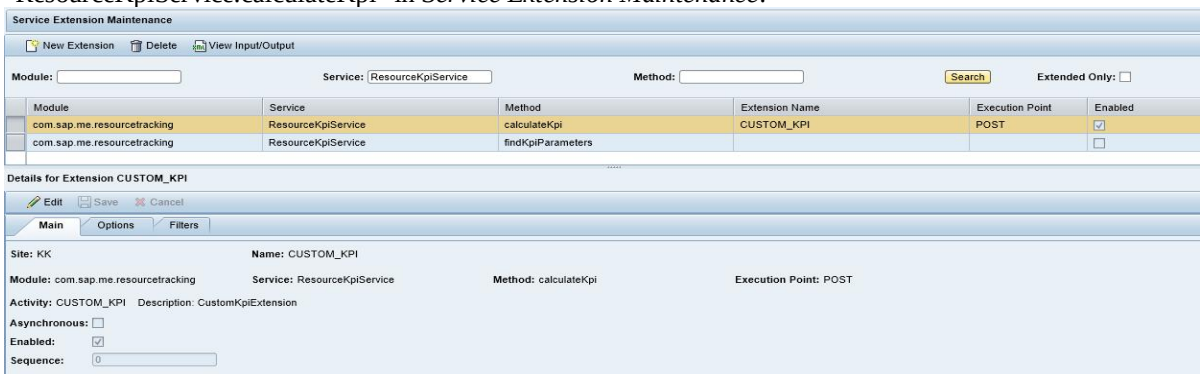
7.1 Activity Maintenance

A custom service extension activity must be created as follows:



7.2 Service Extension Maintenance

The custom service extension must be assigned on "POST" execution point to the API service method "ResourceKpiService.calculateKpi" in *Service Extension Maintenance*:



© Copyright SAP SE 2018

7.3 Service-config.xml

The custom service extension must be configured as a service in service-config.xml:

```
<service name="CustomKpiExtension"
class="com.sap.me.production.impl.CustomKpiExtension" parent="Prototype">
</service>
```

© Copyright SAP SE 2018

7.4 Service Extension Class

The service extension class must extend a generic class ServiceExtension with a parameter type CalculateKpiRequest. Here is an example of a service extension class that implements the same logic as in the standard method ResourceKpiService.calculateKpi:

```
package com.sap.me.production.impl;

import java.math.BigDecimal;
import java.util.List;

import com.sap.me.activity.ActivityRuntimeType;
import com.sap.me.activity.ExecutionType;
import com.sap.me.activity.ServiceExtension;
import com.sap.me.frame.domain.BusinessException;
import com.sap.me.resourcetracking.AvailabilityLoss;
import com.sap.me.resourcetracking.CalculateKpiRequest;
import com.sap.me.resourcetracking.CalculateKpiResponse;
import com.sap.me.resourcetracking.FindKpiParametersResponse;
import com.sap.me.resourcetracking.PlannedStop;

@ExecutionType(ActivityRuntimeType.SYNC)
public class CustomKpiExtension extends ServiceExtension<CalculateKpiRequest>{
        private static final long serialVersionUID = 1L;

        public void execute(CalculateKpiRequest calculateKpiRequest) throws BusinessException {
                CalculateKpiResponse calculateKpiResponse = this.invocationContext.getResult();
                FindKpiParametersResponse findKpiParameters = calculateKpiResponse; //
ResourceKpiService.findKpiParameters has been already executed when executing core method
"calculateKpi"

                List<PlannedStop> plannedStop = calculateKpiResponse.getPlannedStop();
                BigDecimal allPlannedStopsDuration = BigDecimal.ZERO;
                for (PlannedStop plannedStop1 : plannedStop) {
                        allPlannedStopsDuration =
allPlannedStopsDuration.add(plannedStop1.getDuration());
                }

                List<AvailabilityLoss> availabilityLoss =
calculateKpiResponse.getAvailabilityLoss();
                BigDecimal allAvailabilityLossDuration = BigDecimal.ZERO;
                for (AvailabilityLoss availabilityLos : availabilityLoss) {
                        allAvailabilityLossDuration =
allAvailabilityLossDuration.add(availabilityLos.getDuration()).add(availabilityLos.getMinorStoppag
eDuration());
                }

                BigDecimal plantWorkingTime = calculateKpiResponse.getPlantWorkingTime();

                calculateKpiResponse.setPlannedProductionTime(plantWorkingTime.subtract(allPlanne
                dStopsDuration));

                calculateKpiResponse.setOperatingTime(calculateKpiResponse.getPlannedProductionTi
                me().subtract(allAvailabilityLossDuration));

                calculateKpiResponse.setRunningTime(calculateKpiResponse.getShopOrderOperatingTim
                e().subtract(findKpiParameters.getCalculatedSpeedLossDuration()).subtract(findKpi
                Parameters.getMinorStoppageDuration()));

                calculateKpiResponse.setFullyProductiveTime(calculateKpiResponse.getRunningTime()
                .subtract(findKpiParameters.getQualityLoss()));
        }
}
```

© Copyright SAP SE 2018

7.5 OEE KPI Calculation

The controller logic of OEE POD will use the following formula to calculate final KPIs to be shown on UI:

- Availability = 100% * calculateKpiResponse.getOperatingTime() / calculateKpiResponse.getPlannedProductionTime()
- Performance = 100% * calculateKpiResponse.getRunningTime / calculateKpiResponse.getShopOrderOperatingTime()
- Quality = 100% * calculateKpiResponse.getFullyProductiveTime() / calculateKpiResponse.getRunningTime()
- OEE = (Availability * Performance * Quality) / 10000

7.6 OEE Time Model Terminology

The following table provides mapping of used terminology with alternative OEE terminology:

SAP ME OEE Time Model	Alternative OEE Time Model
Plant Working Time	Total Production time
Planned Stop	Scheduled Down
Planned Production Time	Loading Time
Availability Loss	Unscheduled Down
Operating Time	Net Production time
Running Time	Net Operating time
Fully Productive Time	Value Operating time

7.7 FindKpiParametersResponse

This DTO Type is available for the extension service class and contains parameters that can be used in KPI calculations:

- List<PlannedStop> **plannedStop** - Collection of time elements and durations of all scheduled downs aggregated by the time element of the resource reason code;
- List<AvailabilityLoss> **availabilityLoss** - Collection of time elements, durations and minor stops duration of all unscheduled downs aggregated by time element of resource reason code;
- BigDecimal **minorStoppageDuration** - Sum of durations of all minor stoppage statuses with type PERFORMANCE;
- BigDecimal **calculatedSpeedLossDuration** - Speed loss duration as returned by SpeedLossService;
- BigDecimal **qualityLoss** - Time spent on SFCs being scrapped. Calculated as RunningTime * (scrapped quantity / (scrapped quantity + completed quantity);
- BigDecimal **plantWorkingTime** - Plant working time that includes full shift duration except breaks duration;

© Copyright SAP SE 2018

- BigDecimal **shopOrderOperatingTime** - Plant working time regarding to shop order except for breaks and downtimes.

All durations are in milliseconds.

© Copyright SAP SE 2018

7.8 Customization Scenarios

The standard logic does summarization of all scheduled downs in variable `allPlannedStopsDuration`.

```
List<PlannedStop> plannedStop = calculateKpiResponse.getPlannedStop();
                BigDecimal allPlannedStopsDuration = BigDecimal.ZERO;
                for (PlannedStop plannedStop1 : plannedStop) {
                        allPlannedStopsDuration =
allPlannedStopsDuration.add(plannedStop1.getDuration());
                }
```

It is possible to implement some additional logic to filter out scheduled downs by time element type criteria. Collection `calculateKpiResponse.getPlannedStop()` contains durations aggregated by the time element type, each element of the collection has type PlannedStop containing timeElementRef and duration.
So, for example, it's possible to check if timeElementType should be added to the planned stops duration:

```
if (!plannedStop1.getTimeElementRef().equals(excludeTimeElementTypeRef)){
        allPlannedStopsDuration = allPlannedStopsDuration.add(plannedStop1.getDuration());
}
```

A similar customization scenario can be applied to calculation of availability loss – variable `allAvailabilityLossDuration`. Collection `calculateKpiResponse.getAvailabilityLoss()` is also aggregated by time element types and has elements AvailabilityLoss with fields: timeElementRef, duration, minorStoppageDuration.

Also, it's possible to change resulting parameters in `calculateKpiResponse`, for example to include minor stoppages in a different way.

© Copyright SAP SE 2018