DTVT\_BLUE Rule Automation Logic

# Table of Contents

[1. Table of Contents 2](#_Toc482635407)

[2. Introduction 3](#_Toc482635408)

[3. Rules Automation Logic 3](#_Toc482635409)

[3.1. RULE\_DB\_ROUTE\_0003 3](#_Toc482635410)

[3.2. RCD\_SIGNAL\_0001 3](#_Toc482635411)

[3.3. RULE\_TMS\_PLATF\_0003 4](#_Toc482635412)

[3.4. RULE\_TMS\_SDDB\_0013 4](#_Toc482635413)

[3.5. RULE\_TMS\_SIGNAL\_0012 5](#_Toc482635414)

[3.6. RULE\_TMS\_SSP\_0012 5](#_Toc482635415)

[3.7. RULE\_DB\_ROUTE\_0001 6](#_Toc482635416)

[3.8. RCD\_DB\_SIGAREA\_0003 6](#_Toc482635417)

[3.9. RCD\_TMS\_RMZ\_016 7](#_Toc482635418)

[3.10. RULE\_TMS\_STABL\_0003 8](#_Toc482635419)

# Introduction

This document describes the design logic to automate the verification rules using the tool DTVT\_BLUE.

The logic is described by the verification leader for individual rules specified in GfPD document. Verification leaser needs to specify the various input tables, constants needed to verify the rule. Rule automation logic shall be used by the developer to write the script.

# Rules Automation Logic

## RULE\_DB\_ROUTE\_0003

### Input Table

* Routes\_Cap
* Switchs\_Cap
* Points\_Cap
* Secondary\_Detection\_Devices\_Cap
* Signalisation\_Areas\_Cap

### Constants

None

### Description

1. Get all routes from the table Routes\_Cap
2. For each route get the ‘Switch\_ID\_List’
3. For each switch in the ‘Switch\_ID\_List’ get the list of points from the columns ‘Convergent\_Point\_ID\_List’ and ‘Divergent\_Point\_ID\_List’
4. For each points in the point list get associated SDD from the table ‘Secondary\_Detection\_Devices\_Cap’ and create Switch\_SDD\_List
5. For each SDD in the Switch\_SDD\_List get the associated Signalisation Area Name of type ‘CBI’ from ‘Signalisation\_Areas\_Cap’ and prepare a list of ‘CBI\_Signalisatin\_Area\_List’. This list contains signalization area name associated to all switches of the ‘Switch\_ID\_List’
6. ‘CBI\_Signalisatin\_Area\_List’ represent a list of for all Switches in the route selected in step (ii)
7. Now if all items in the list ‘CBI\_Signalisatin\_Area\_List’ are same then test case is ‘OK’ else test case is ‘NOK’

## RCD\_SIGNAL\_0001

### Input Table

* Signals\_Cap
* Service\_Stopping\_Points\_Cap

### Constants

None

### Description

1. Get SSP\_list from Service\_Stopping\_Points\_Cap
2. For each SSP in SSP\_list get the SSP\_Kp Associated Signal from Signals\_Cap
3. Get the Signal\_Kp
4. The distance between SSP\_Kp and Signal\_Kp >= 200 cm as specified by the rule

## RULE\_TMS\_PLATF\_0003

### Input Table

* Platforms\_cap

### Constants

* C\_Late\_Change\_Distance the list of constant for all platforms has to be provided by the Verification Leader in the format specified by input file ‘C\_Late\_Change\_Distance.xlsx’.

|  |  |
| --- | --- |
| **Platform** | **C\_Late\_Change\_Distance (cm)** |
| PF\_ASW\_3B\_LU | 51489 |
| PF\_ASW\_4B\_RD | 52078 |

### Description

1. Get platform\_list and associated list of C\_Late\_Change\_Distance from Platforms\_cap
2. For each platform in platform\_list the value of its constance C\_Late\_Change\_Distance should match with the value mentioned in the file C\_Late\_Change\_Distance.xlsx’

## RULE\_TMS\_SDDB\_0013

### Input Table

* Signals\_Cap
* SDDB\_Cap
* Tracks\_Cap

### Constants

None

### Description

1. Get Track\_List from Tracks\_Cap
2. For each track get the signal\_list of signals of type ‘Route or Spacing’
3. In signal\_list separate the signals in ‘Up’ and ‘Down’ direction in two diferent list ie. Signal\_up and Signal\_dn
4. For Signal\_up and Signal\_dn, between every pair of successive signals count the total number of SDDBs
5. If count of SDDB between any two signal in same direction of same track is equal to ‘Zero’ than test fails except Signal Kp is not on SDDB boundary

## RULE\_TMS\_SIGNAL\_0012

### Input Table

Test uses SyDB. From SyDB main nodes used:

Routes,Signals,Switchs,Points,Blocks

### Constants

None

### Description

1. Get the list of Route\_List from SyDB
2. For each Route in Route\_List get the Origin\_Signal
3. Get Kp value of Origin\_Signal , Sig\_Kp = 'Value' + 'Corrected\_Gap\_Value' + 'Corrected\_Trolley\_Value'
4. Using Sig\_Kp get associated Block\_ID of Origin\_Signal
5. For Route get Switch\_ID\_List
6. For each switch, get associated Points\_List
7. For each Point in Points\_List get list ‘Deadlocking\_Block\_ID\_List’
8. If Block\_ID of Origin\_Signal is in the lists of ‘Deadlocking\_Block\_ID\_List’ then test Fails

## RULE\_TMS\_SSP\_0012

### Input Table

* Service\_Stopping\_Points\_Cap
* TRFC\_Cap
* Lines\_Cap

### Constants

Line.Min\_Head\_Opposite\_Shunting\_Axle\_Vehicle\_Length

Local\_D\_Joint

Loc\_Error

Stopping\_accuracy

### Description

1. Get the list of SSP from Service\_Stopping\_Points\_Cap
2. Get the Kp Values of SSP ( = KpValue + KpCorrected\_Trolley\_Value)
3. Get the Train\_Formation\_Characteristics\_ID\_List from Service\_Stopping\_Points\_Cap for SSP
4. Get the distance of closest rear SDDB upstream the train ( Dssp\_sddb)
5. Get the Formation\_Length (Dtrain ) from TRFC\_Cap.csv
6. For each Train specifid in Train\_Formation\_Characteristics\_ID\_List, get the train length Dtrain

Tail of the train when stopped at the SSP) < Line.Min\_Head\_Opposite\_Shunting\_Axle\_Vehicle\_Length - Local\_D\_Joint - Loc\_Error - Stopping\_accuracy.

**(Dssp\_sddb  - Dtrain ) < (**Line.Min\_Head\_Opposite\_Shunting\_Axle\_Vehicle\_Length - Local\_D\_Joint - Loc\_Error - Stopping\_accuracy)

## RULE\_DB\_ROUTE\_0001

### Input Table

* Routes\_Cap
* Signals\_Cap
* Signalisation\_Areas\_Cap

### Constants

None

### Description

* Get Routes from Routes\_Cap
* For each Route in Routes get Origin\_Signal\_ID
* From Signals\_Cap, for Origin\_Signal\_ID get Secondary\_Detection\_Device\_ID
* If Secondary\_Detection\_Device\_ID exist then verify that it belongs to a CBI Signalisation Area listed in Signalisation\_Areas\_Cap

## RCD\_DB\_SIGAREA\_0003

### Input Table

* ZCs\_Cap
* Signalisation\_Areas\_Cap
* Points\_Cap

### Constants

Head\_Correction

NIAP\_Tail\_Correction

### Description

* Get list of ZC Name and ZC\_Area\_ID from ZCs\_Cap
* For each ZC get the SDD list from Signalisation\_Areas\_Cap
* Get the ZC Boundaries by sorting SDD list
* For each boundary of ZC on either side within the distance of max (Head\_Correction, NIAP\_Tail\_Correction ) there must not be any point Kp\_Toe

## RCD\_TMS\_RMZ\_016

### Input Table

* Reverse\_Movement\_Zones\_Cap
* SDDB\_Cap
* TRFC\_Cap

### Constants

None

### Description

To determine whether any SDDB exist between RMZ Begin and End boundaries for the give track:

Get the list of RMZ

* For each RMZ get ‘RMZ\_Type’ , Kp\_Begin, Kp\_End ,Track\_ID and Direction
* From SDDB\_Cap get the list of SDDBs for the same Track\_ID and its Kp Value
* Sort the SDDBs based on the Kp value
* For RMZ of type ‘RMR\_Per\_Zones’ there must not be any SDDB betwenn Kp\_Begin & Kp\_End of RMZ
* For RMZ of type ‘Reverse\_Jog’, “ lengthe of RMZ ( Kp\_Begin ~ Kp End) + Train Fomration Length “ must be contained by two **consecutive** SDDBs



## RULE\_TMS\_STABL\_0003

### Input Table

* Stablings\_Location\_Cap
* TRFC\_Cap
* Service\_Stopping\_Points\_Cap

### Constants

None

### Description

Stablings specified for a stabling location, are in a group and defined back to back. For the stabling location shown below

STL\_HDS\_101B

STL\_HDS\_101A

STL\_HDS\_101C

STL\_HDS\_101D

UP

SSP1

SSP2

SSP3

SSP4

L1

L2

L3

L4

In Up direction for SSP1 trains associated should have length less than L1

For SSP2 train associated should be of length <L2

For SSP3 train associated should be of length <L3

For SSP4 train associated should be of length <L4

For each stabling location get the ordered list of stabling in increasing KP values e.g. [STL\_HDS\_101A

, STL\_HDS\_101B, STL\_HDS\_101C, STL\_HDS\_101D] and test that the Kp End of one stabling is same as Kp Begin of next stabling

Now if Increasing Kp is defined as UP direction then

Get all SSP of stabling defined in “Down’ direction then:

For ssp in ssp\_list\_dn :

get stabling (ssp) and its position in the stabling list , from the list find the length of the stabling space available for this ssp e.g. for SSP2 the stabling will be STL\_HDS\_101B and length available will be L2

for ssp find the train formation list from Service\_Stopping\_Points\_Cap. For every train in the list train \_length should be less than the available length for the respective ssp