Contents

[1. Introduction 3](#_Toc420330642)

[2. Understanding DBTool Work flow 4](#_Toc420330643)

[Step 0 4](#_Toc420330644)

[Step 1 4](#_Toc420330645)

[Step 2 4](#_Toc420330646)

[Step 3 4](#_Toc420330647)

[Step 4 4](#_Toc420330648)

[Step 5 5](#_Toc420330649)

[Step 6 5](#_Toc420330650)

[1. Running DBTool 8](#_Toc420330651)

[2. Viewing XML files 8](#_Toc420330652)

[1. Icons explained 8](#_Toc420330653)

[2. Supported XML file types 9](#_Toc420330654)

[3. Parallel Work Tabs 10](#_Toc420330655)

[4. Edit XML files 11](#_Toc420330656)

[5. View diff between XML files 12](#_Toc420330657)

[1. What do the colours mean? 12](#_Toc420330658)

[6. Merge Diff between XML files 13](#_Toc420330659)

[1. Synchronise 13](#_Toc420330660)

[2. Drag-Drop 14](#_Toc420330661)

[7. Import Schema from BBF XML 16](#_Toc420330662)

[8. Convert Schema to Instance XML file 17](#_Toc420330663)

[9. How to Leverage the Custom Menu 18](#_Toc420330664)

[10. Model Generation 19](#_Toc420330665)

[11. Perform Bulk edits using Excel 20](#_Toc420330666)

[12. Import/Export between XML & Excel files 21](#_Toc420330667)

[1. Excel to XML conversion 21](#_Toc420330668)

[2. Excel to XML conversion 21](#_Toc420330669)

[3. Working with Schema files – New, Diff, Merge, View & Edit 22](#_Toc420330670)

[4. Working with Instance files – New, Diff, Merge, View & Edit 23](#_Toc420330671)

[5. Working with Model files – New, Diff, Merge, View & Edit 24](#_Toc420330672)

[6. DBTool – Under the hood 25](#_Toc420330673)

[1. Configuring paths 25](#_Toc420330674)

[2. Fiddling with marker colours 25](#_Toc420330675)

[3. Window resizing 25](#_Toc420330676)

[4. Custom menus 25](#_Toc420330677)

[5. Snip out Services right at pre-processing stage 25](#_Toc420330678)

[6. Modifying code directory paths 25](#_Toc420330679)

[7. Watermarking 25](#_Toc420330680)

[8. Merging multiple control XMLs 25](#_Toc420330681)

[7. Header Generation module invocation 27](#_Toc420330682)

[8. The Good Old Backend 28](#_Toc420330683)

# Introduction

01\_Install\_Guide.docx – Please refer to all the steps mentioned in document and install the DBTool if you have not done already!

This guide shall explain the basic steps in using DBTool and performing various bulk and individual operations on XML files.

For the rest of the document, DBTool shall refer to the operations from GUI only. The operations supported from Backend shall continue to exist, however performing the same operations from GUI is highly recommended.

# Understanding DBTool Work flow

The DBTool Work flow is summarized in Four easy steps below.

## Step 0

Step 0 is typically done by Project Manager. As a developer you need not do this step. The latest BBF XMLs gets converted and ready for reference by every subsystem. You would be notified when to re-base.

## Step 1

You would work on your Control XML files in DBTool by doing a *File->Open.* Adding/Deleting/Modifying Objects, Parameters, Attributes and Values happen here.

You may also want to do a *Schema->Merge* if you wish to do a mash up of multiple base services from BBF and prepare a custom service module.

## Step 2

Once your Control file is prepared, it is time to prepare the corresponding data Instance files. Use Instance->Create to do it.

## Step 3

Here you step out of the comforts of the FrontEnd. Please invoke *genHdr* to generate the Schema-Id file for your module.

You also generate the header file for your module at this stage.

## Step 4

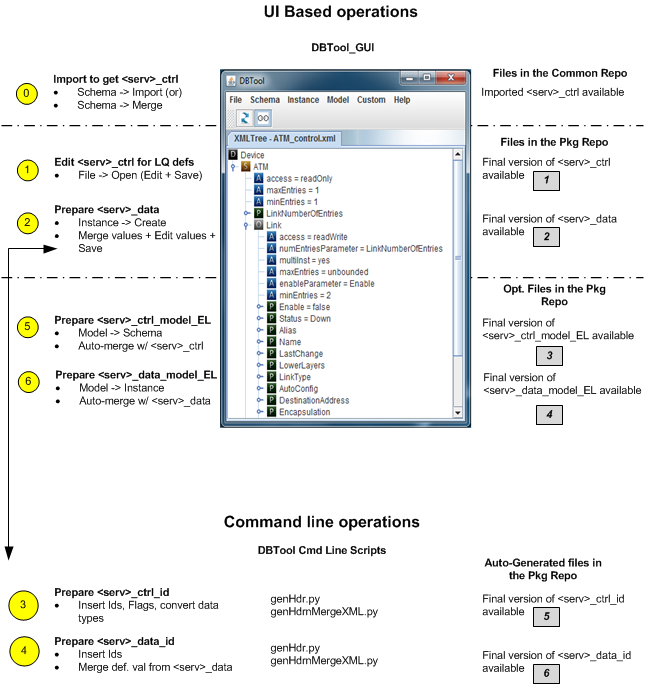
Again please invoke *genHdr* to generate the Instance-Id file for your module.



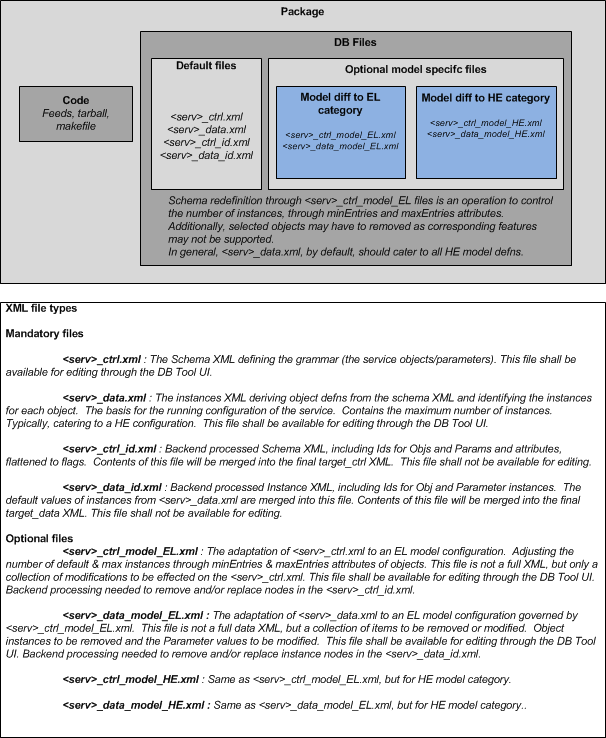
## Step 5

Step 5 is being touched upon only in the diagram below. You need to prepare the Entry Level Schema model definition for your module, if you intend to have an Entry Level subset of features for your module.

## Step 6

Step 6 is being touched upon only in the diagram below. You need to prepare the corresponding Entry Level Instance model definition for your module, if you intend to have an Entry Level subset of features for your module. 

Finally the **Schema, Instance, schema-Id, instance-Id** and **header** files for your module needs to be checked in.

****

# Running DBTool

* **For Windows:**

Browse to  ***DBTool/src***

Double click and run ***dbTool.bat***

* **For Linux:**

Please have a graphical session (NX, VNC, X11 etc) open before you run this.

***# cd DBTool/src***

***# ./dbTool.sh***

# Viewing XML files

Perform: ‘File->Open’ and browse to the specific file/s that you wish to open and view.

If you select and open multiple XML files, each file will be independently opened in a separate Work Tab

However if you open the same file in multiple tabs intending to perform separate operations, the behavior is undefined.

## Icons explained

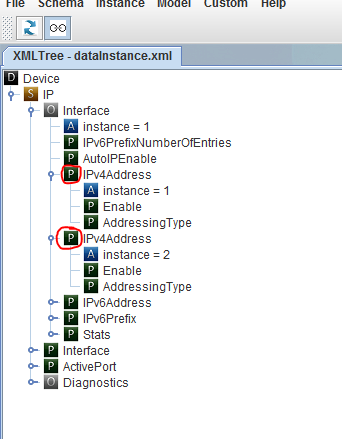
D – Symbolises a node at Device level.

S – Symbolises a node at a Service level

O – Symbolises an object or a sub-object node where ever applicable

A – Symbolises an Attribute node where ever applicable

P – Symbolises a Parameter node where ever applicable



## Supported XML file types

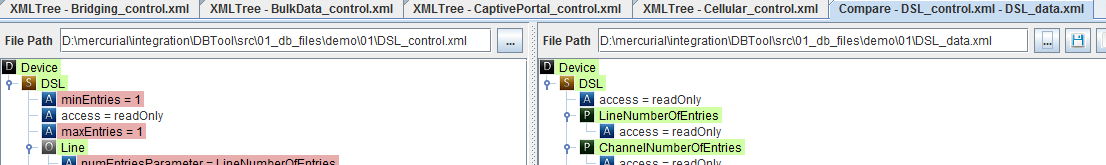
Schema, Instance, Model and BBF XML files are supported by default. The edit capabilities of each is context sensitive and is not being captured here.

Any generic XML file may also be loaded and viewed but its operation is not supported.

# Parallel Work Tabs

You may open multiple work tabs in parallel. Some performing comparison, merge, edit or view operations.

Opening same file twice in separate tabs is not recommended and its behavior is undefined (for now).



# Edit XML files

<TBD>

# View diff between XML files

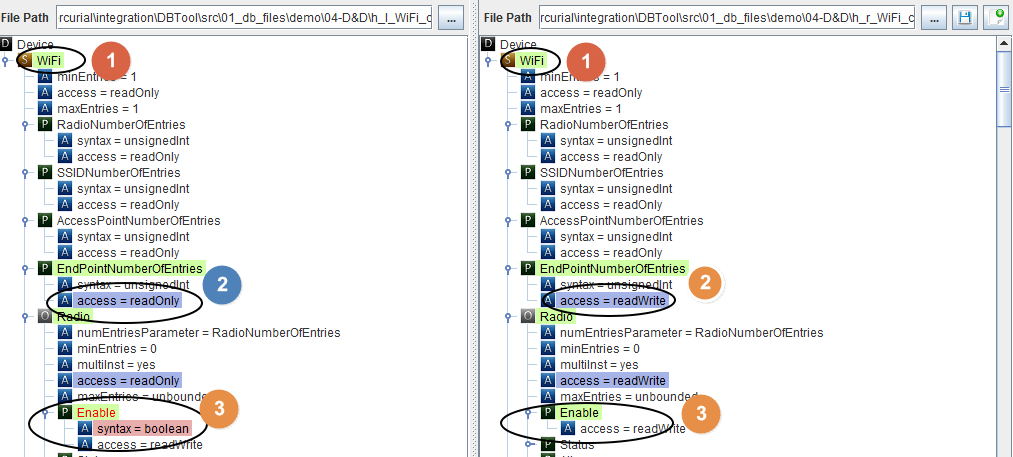
You can view the diff between two Schema, Instance or Model files.

Steps are either of:

* Schema->Merge
* Instance-> Merge
* Model->Merge

## What do the colours mean?

* Guiding colour - In below figure, the green colour shown near (1) indicates that there are some diffs underneath in the hierarchy. The green colour disappears when all the diffs underneath it have been addressed.
* Mismatch colour – The blue colour shown near (2) denotes that there are some mismatch in the specific value. This is analogous to a merge conflict.
* Missing colour - The red colour shown near (3) denotes that the specific node is missing on the other side.



# Merge Diff between XML files

In DBTool, the RHS is editable and LHS is maintained as read-only. There are two ways in which a diff can be managed across two XML files.

## Synchronise

You may right click at a missing node (red colour) or at a mismatched node (blue colour) and select Synchronise option to have it reflected at RHS.



.



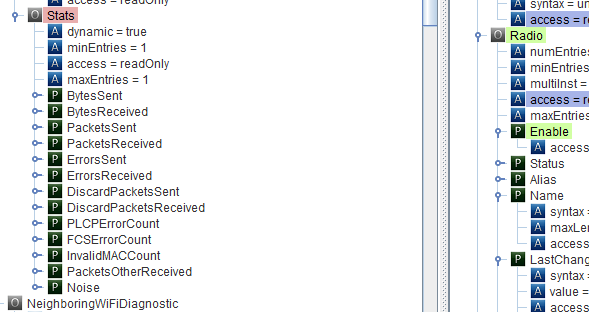
You can also perform a Sync at an Object level where in all the child nodes below that object are also synchronized automatically.

## Drag-Drop

To initiate a Drag-drop operation, decouple the LHS and RHS panes to activate independent scrolling bars. This can be done by clicking the link/unlink button.

Change state from  to .

After this is done, you can drag and drop a node (or a sub-tree) from LHS to the corresponding parent or to any independent position on the RHS.



In this figure, LHS shows Device.Wifi.Radio.Stats being dragged and dropped over Device.Wifi.Radio.

# Import Schema from BBF XML

[Important]This operation should be performed only after authorization from Build / Release management!

This operation will overwrite the service XML files and this may not be what you want.

In case this is what you want, the steps are: File->Schema->Import. Browse to the BBF XML file (TR-181, TR-104, TR-98 etc) and select it.

This is an intensive operation and could potentially take approximately two to three minutes. Please do not make any parallel edits during this time.

# Convert Schema to Instance XML file

<TBD>

# How to Leverage the Custom Menu

You can use the Custom Menu to execute your own (simple) bash commands or scripts as a pre or post command.

Edit *‘config.properties’* file and make an additional menu entry or modify an existing one.

(MenuName\* and MenuCmd\* are to be touched)

# Model Generation

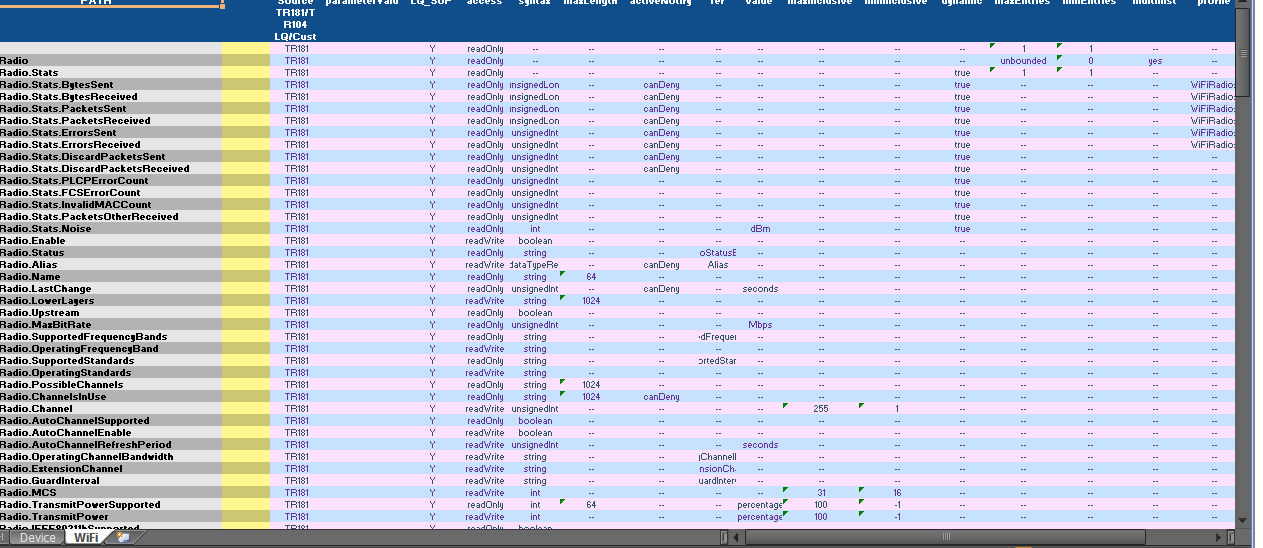
<TBD>

# Perform Bulk edits using Excel

[Important] Use this feature only if you know what you are doing!

You can add new parameters, objects, attributes or change any existing entries using Excel representation of the XML file. Multiple tabs are also supported.

Please note, there is no strict syntax checking performed while converting the Excel back to XML file. You might land with a corrupt / inconsistent XML file, if you do not know what you are doing with the Excel file!



# Import/Export between XML & Excel files

## Excel to XML conversion

**File -> Import from Excel** shall allow **compatible** XLSX file to be imported / converted to XML format.

The destination path for the generated XML file is fixed at: DBTool\src\01\_db\_files\xml\_files\...

## Excel to XML conversion

**File -> Export to Excel** shall allow **compatible** XML file to be imported / converted to Excel format.

The destination path for the generated XML file is fixed at: DBTool\src\01\_db\_files\excel\_files\...

# Working with Schema files – New, Diff, Merge, View & Edit

# Working with Instance files – New, Diff, Merge, View & Edit

<TBD>

# Working with Model files – New, Diff, Merge, View & Edit

# DBTool – Under the hood

## Configuring paths

File: config.properties

Edit *‘defaultlocation’, ‘scriptsdir’* and *‘servicedir’* to reflect the location of DBTool in your installation.

## Fiddling with marker colours

File: colors.properties

Modify the hex colours mentioned in this file, if you wish to change the diff or marker colours to your preference.

## Window resizing

File: config.properties

Edit *screenWidth* and *screenHeight* to reflect the display window size that suits you.

## Custom menus

File: config.properties

Refer ‘How to Leverage the Custom Menu’ for details

## Snip out Services right at pre-processing stage

File: skipList.properties

Add service level entries here if you wish to snip out the unsupported services for your release. (Warning: Keep Build / Release Management in loop before you touch this file.)

## Modifying code directory paths

File: common.py

Modify entries in this path to change the code directory structure.

## Watermarking

File: waterMark.properties

Schema, Instance and Model files are watermarked automatically. Editing this file helps you change the watermarking text, if needed.

## Merging multiple control XMLs

File: mergeServices.properties

Editing entries in this file help you merge multiple BBF services under a custom Service name.

For example, *IP\_control.xml, Bridging\_control.xml,* and *PPP\_control.xml* are merged and a custom *Network\_control.xml* is generated by configuring this section/file.

# Header Generation module invocation

<TBD>

# The Good Old Backend

This is a straightforward and deep dive HOWTO guide that specifically addresses certain use cases for developers.

Notes & Disclaimers:

* Do not change the naming conventions for the Excel and XML files described below. The tool will not run if you do not follow the correct file naming conventions mentioned below.
* Do not make experimental changes in the Excel files, the parsing is NOT stringent in the convertor.

Pre-configured locations for various database files:

1. ..\..\ 01\_db\_files\xml\_files\02\_developer

* Contains master TR-181 and TR-141 xml files

1. ..\..\ 01\_db\_files\excel\_files\02\_developer

* Contains master TR-181 and TR-141 Excel files

1. ..\..\ 01\_db\_files\xml\_files\02\_developer\service\_level

* Contains service level xml files for developers use

1. ..\..\ 01\_db\_files\excel\_files\02\_developer\service\_level

* Contains service level Excel files for developers use

How to set environment variables?

* 1. Go to **DBTool/02\_src/utils/**
  2. Execute batch file **setPath.bat** (if on Windows);

Execute bash script **setPath.sh** (Linux/Unix)

How to Generate Lantiq Control XML from TR-181/TR-104 Base XML?

As a developer you DO NOT have to worry about this. Still if you need to, just run this script:

1. Go to **DBTool/02\_src/utils/**
2. Execute **python genCtrlXML.py**

With that the current version of TR181/TR-104 XML files gets converted to Lantiq recommended format.

The consolidated Lantiq Control XML file is generated at the following location:

**DBTool/01\_db\_files/xml\_files/02\_developer/tr\_181\_lq\_control.xml**

We repeat, as a developer you do not have to fire this script. Perform this operation with authorization only!

How to split Lantiq Control XML to individual Service Level XMLs?

As a developer you DO NOT have to worry about this. Still if you need to, just run the same script:

Go to **DBTool/02\_src/utils/**

**python genCtrlXML.py**

This would automatically have created the service level XMLs at the following location:

**DBTool/01\_db\_files/xml\_files/02\_developer/service\_level**

We repeat, as a developer you do not have to fire this script. Perform this operation with authorization only!

Why to convert Service Level XML file to Service Level Excel files (and back)?

As a module owner / developer you own your service Control & Data XML file. But if working with XML files is not your forte, the tool will convert this to Excel format for you. You can work in Excel, and the tool will help you convert this back to XML. You would need to check in your Service level Control & Data XMLs in at the location (specified by SD Manager). The Excel files are NOT intended to be checked in.

The tool will help you convert both Control & Data XML file to Excel and back.

**Warning**: The tool will not do any intelligent parsing for the conversion. So please be careful with the changes you make in Excel file.

How to convert Service Level Control XML files to Service Level Control Excel files?

As a module developer, YES, you do need to do this.

Go to **DBTool/02\_src/utils/**

**python x2x.py –toEXL <file\_name>.xml**

For example if you own the module for Wi-Fi, this is how it would look:

**python x2x.py –toEXL WiFi\_control.xml**

*(The path to file name is not being shown above to reduce visual clutter. You would find the file here:*

*DBTool\01\_db\_files\xml\_files\02\_developer\service\_level)*

This generates the corresponding Excel file at this location:

**DBTool\01\_db\_files\excel\_files\02\_developer\service\_level**

So if you owned Wi-Fi module, your corresponding Excel file is:

DBTool\01\_db\_files\excel\_files\02\_developer\service\_level\**WiFi\_control.xlsx**

Please work on the additions, modifications, deletions for your module’s Excel file. Save it. Do not rename it.

**Note: DO NOT check in the Excel file.**

After working on the Excel file, you need to convert it back to XML file and then check it in. Refer the below steps.

How to convert Service Level Control Excel files to Service Level Control XML files?

As a module developer, YES, you do need to do this.

Go to **DBTool/02\_src/utils/**

**python x2x.py –toXML <file\_name>. xlsx**

For example if you own the module for Wi-Fi, this is how it would look:

**python x2x.py –toXML WiFi\_control.xlsx**

*(The path to file name is not being shown above to reduce visual clutter. You would find the file here:*

*DBTool\01\_db\_files\excel\_files\02\_developer\service\_level)*

This generates the corresponding XML file at this location:

**DBTool\01\_db\_files\xml\_files\02\_developer\service\_level**

So if you owned Wi-Fi module, your corresponding XML file is:

DBTool\01\_db\_files\xml\_files\02\_developer\service\_level\**WiFi\_control.xml**

Note: the tool has now overwritten the Control (or Data)

How to convert Service Control XML to Service Data XML?

As a developer, do I need to do this? Mostly Yes! But talk to SD Manager if you need to do this, or has this been done for your module.

Go to **DBTool/02\_src/utils/**

**python c2d.py <service>\_control.xml**

(Path to <service>\_control.xml is mentioned above)

Corresponding Service level Data XML is created at same directory as the Service Control XML.

Eg. If you own the Wi-Fi module, executing

**python c2d.py WiFi\_control.xml**

will result in **WiFi\_data.xml** file being created at this location:

**DBTool\01\_db\_files\xml\_files\02\_developer\service\_level**

(You can follow steps mentioned below to work on the Service Data Excel file)

How to convert Service Level Data XML files to Service Level Data Excel files?

Same process as detailed for Control XML – except that the file name convention for data is:

<service>\_data.XML (in lieu of <service>\_control.XML)

<service>\_data.XLSX (in lieu of <service>\_control.XLSX)

How to convert Service Level Control Excel files to Service Level Control XML files?

Same process as detailed for Control Excel – except that the file name convention for data is:

<service>\_data.XML (in lieu of <service>\_control.XML)

<service>\_data.XLSX (in lieu of <service>\_control.XLSX)

How to clean up intermediate files?

The script name is: cleanup.py (Usage is reserved for now – has dangerous repercussions)