

# Quick Start Guide

For  
**openCONFIGURATOR**

Prepared By  
**Kalycito Infotech Pvt Ltd.,**  
India

<b>Identifier</b>	Quick_Start_Guide	<b>Version</b>	1.03
<b>Prepared By</b>	Ramakrishnan P	<b>Date</b>	13-Sep-2013
<b>Approved By</b>	Vinod PA	<b>Confidentiality</b>	Public domain document

## Revision History

Version	Date	Modified By	Remarks
0.01	15-Apr-2009	Kalycito Powerlink Team	Initial Draft
1.00	18-May-2009	Kalycito Powerlink Team	Finalized and has cosmetic changes
1.01	29-Apr-2013	Ramakrishnan P	Updated the guide with the new features and major changes to the document for the openCONFIGURATOR version 1.3.0
1.02	05-Jul-2013	Ramakrishnan P	Minor changes and updated section 3.4
1.03	13-Sep-2013	Ramakrishnan P	Formatting update.

## Table of Contents

1 Introduction.....	5
1.1 Context.....	5
1.2 Scope.....	5
1.3 References.....	5
2 Setup.....	6
3 Sample project.....	7
3.1 Creating a sample project.....	7
3.2 Add a CN.....	11
3.3 Adding process variable to openPOWERLINK CN.....	14
3.4 PDO Mapping for the CN process variables.....	21
3.5 Build the sample project.....	25
4 Conclusion.....	30
5 Appendix – Abbreviations.....	31
6 Support.....	32

## Illustration Index

Illustration 1: New Project Menu.....	7
Illustration 2: Create New Project.....	7
Illustration 3: Project Wizard - Name.....	8
Illustration 4: Project Wizard – MN XDD.....	9
Illustration 5: MN Node Created.....	10
Illustration 6: Add CN Menu.....	11
Illustration 7: Add CN Window.....	12
Illustration 8: CN Created.....	12
Illustration 9: View Menu.....	13
Illustration 10: Add Index Menu.....	14
Illustration 11: Add Index Window.....	14
Illustration 12: Index Added - Tree.....	15
Illustration 13: Index Properties - Empty.....	16
Illustration 14: Index Added.....	17
Illustration 15: Add SubIndex Menu.....	18
Illustration 16: Add SubIndex Window.....	18
Illustration 17: Add SubIndex Properties- 1.....	19
Illustration 18: Add SubIndex Properties- 2.....	19
Illustration 19: Input PI variables.....	20
Illustration 20: PDO mapping table.....	21
Illustration 21: PDO mapping table - Select Node id.....	22
Illustration 22: PDO mapping table - Select Index Id.....	22
Illustration 23: PDO mapping table - Select SubIndex Id.....	22
Illustration 24: PDO mapping table - Select Length.....	23
Illustration 25: PDO mapping table - Offset.....	23
Illustration 26: PDO mapping table - RPDO mapping values.....	24
Illustration 27: Build Project Menu.....	25
Illustration 28: Build Project - Auto Generate.....	25
Illustration 29: cdc_xap Folder View.....	26

# 1 Introduction

## 1.1 Context

The purpose of this document is to help users setup openCONFIGURATOR, quickly create and edit a sample network configuration project with a CN having one RPDO and one TPDO.

## 1.2 Scope

This document limits its scope with explaining how to create the demo project using openCONFIGURATOR.

## 1.3 References

- openCONFIGURATOR user manual v1.05.
- Ethernet POWERLINK Communication Profile Specification 301 Version 1.1.0.

## 2 Setup

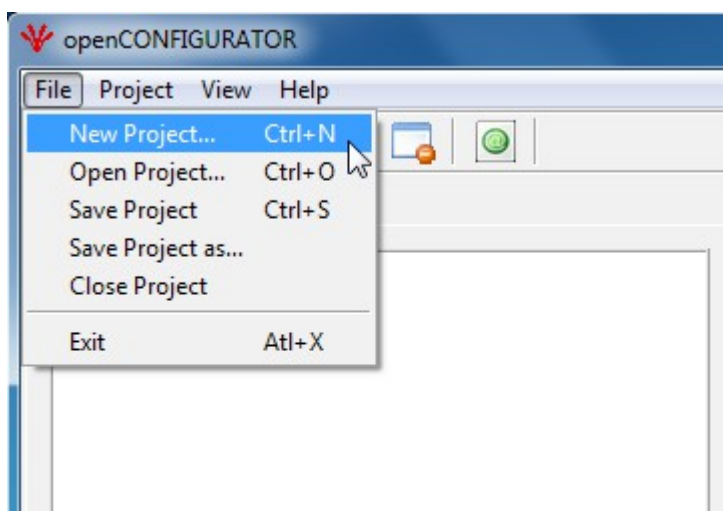
Download latest version of openCONFIGURATOR from <http://sourceforge.net/projects/openconf/> and refer openCONFIGURATOR user manual v1.05 for installation instructions.

### 3 Sample project

This sample project will be used to create a network configuration having one CN with one byte TPDO & one byte RPDO mapped to the MN.

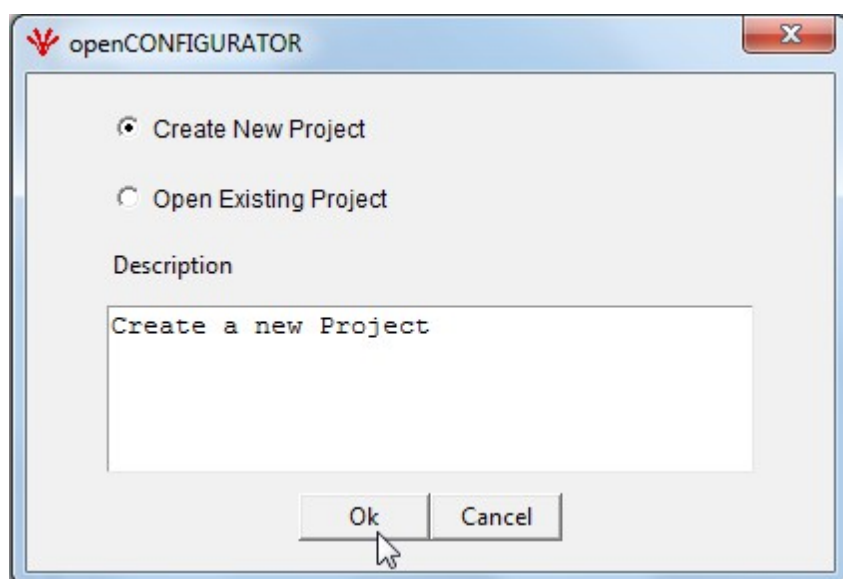
#### 3.1 Creating a sample project

The users can create a new project by selecting the 'File > New Project' as shown in Illustration 1: New Project Menu or using the keyboard shortcut 'CTRL + N'.



*Illustration 1: New Project Menu*

To create a project, choose 'Create New Project' option and click 'Ok' as shown in Illustration 2: Create New Project.



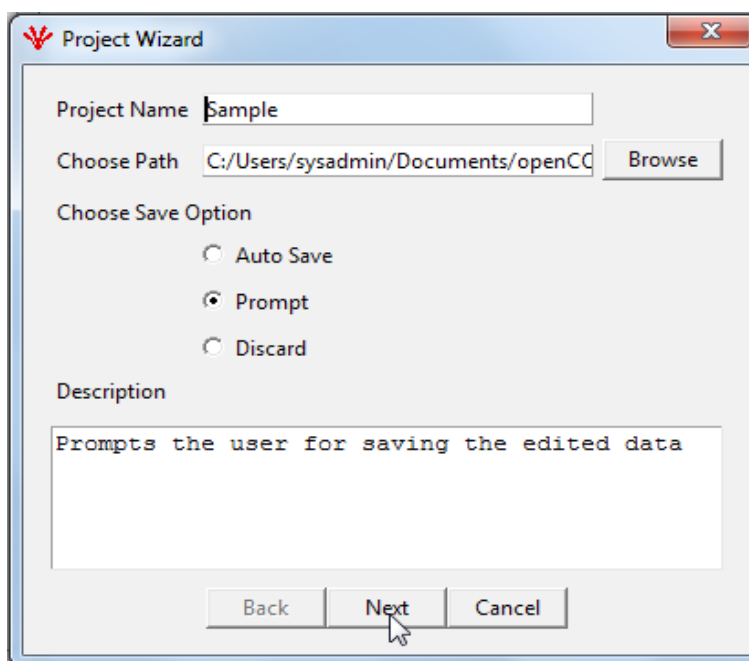
*Illustration 2: Create New Project*

After clicking on the 'Ok' button the tool will guide you through a project creation wizard:

- Set the 'Project Name' for the project.
- Choose the project location by clicking on 'Browse'.
- Choose the 'Save' option as required.

Save option	Description
AutoSave	Saves the configuration automatically without prompting the user
Prompt	Prompts the user for changes to be saved or not
Discard	Discards any modifications made to the configuration

In this sample project, enter 'Sample' as the 'Project Name' and select 'Prompt' as the save option and click 'Next' as shown in Illustration 3: Project Wizard - Name.



*Illustration 3: Project Wizard - Name*



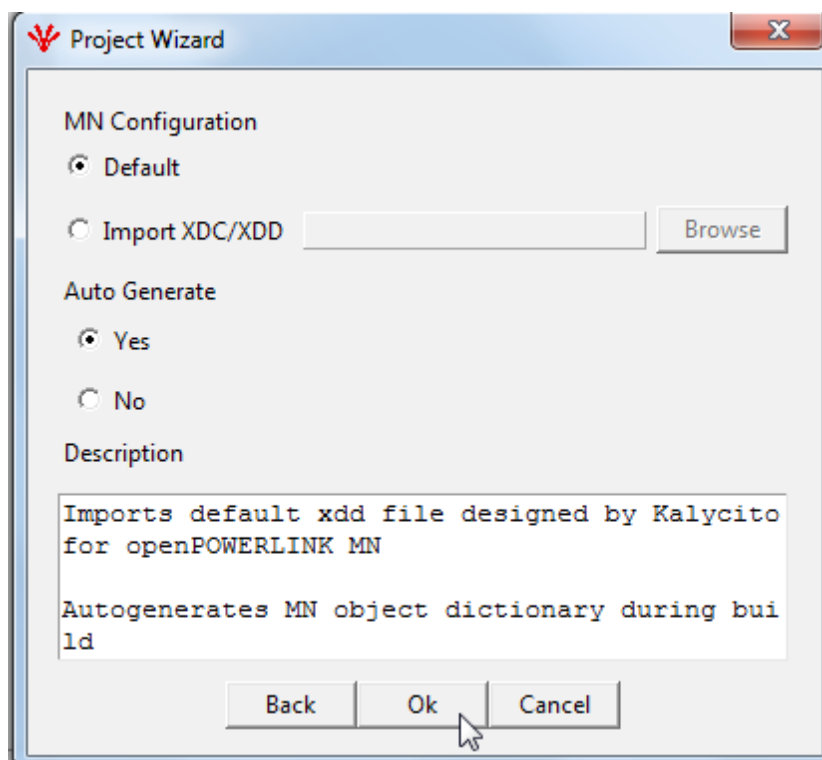
After clicking on the “Next” button the tool will guide you through the next step of MN configuration.

Configuration Option	Description
Default	Default MN xdd which will be available with the installation package
Import XDD/XDC	User defined MN configuration

Select the relevant 'Auto Generate' option.

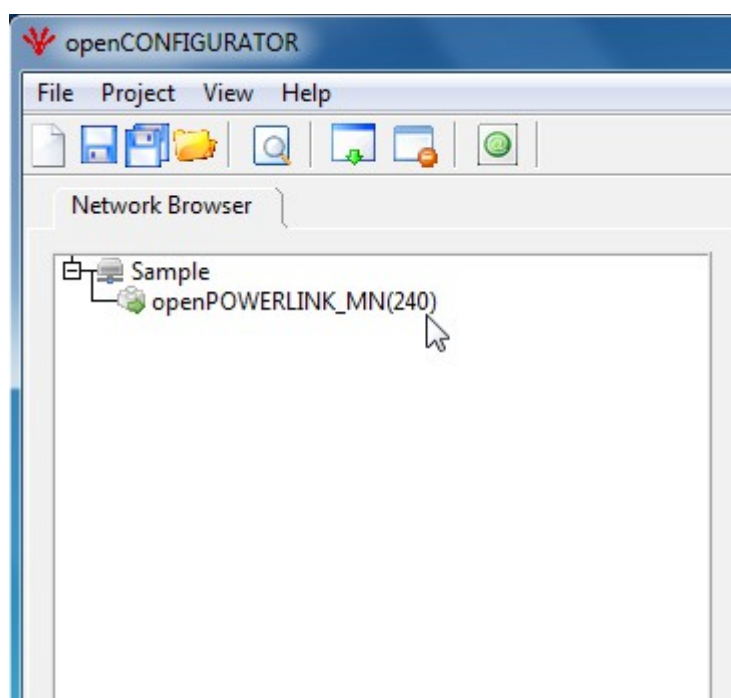
Auto Generate Option	Description
Yes	The MN configuration will be auto generated with the available CN's configuration
No	The MN configuration will have to be manually generated/updated by the user

In this sample project, select the 'default' MN XDD and 'Yes' for Auto Generate options and then click 'Ok' as shown in Illustration 4: Project Wizard – MN XDD.



*Illustration 4: Project Wizard – MN XDD*

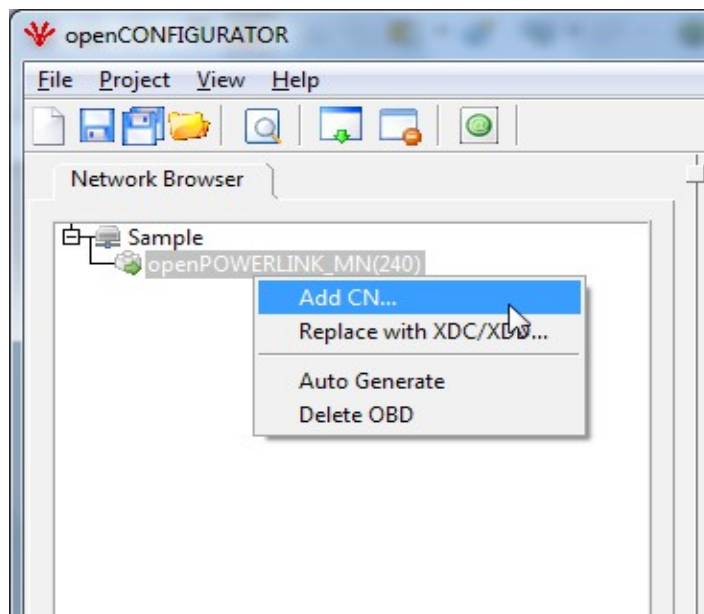
Now the MN node will be created and the project window will look similar to Illustration 5: MN Node Created.



*Illustration 5: MN Node Created*

### 3.2 Add a CN

To add a Controlled Node, right click on openPOWERLINK\_MN in the tree browser and choose 'Add CN' option as shown in Illustration 6: Add CN Menu.

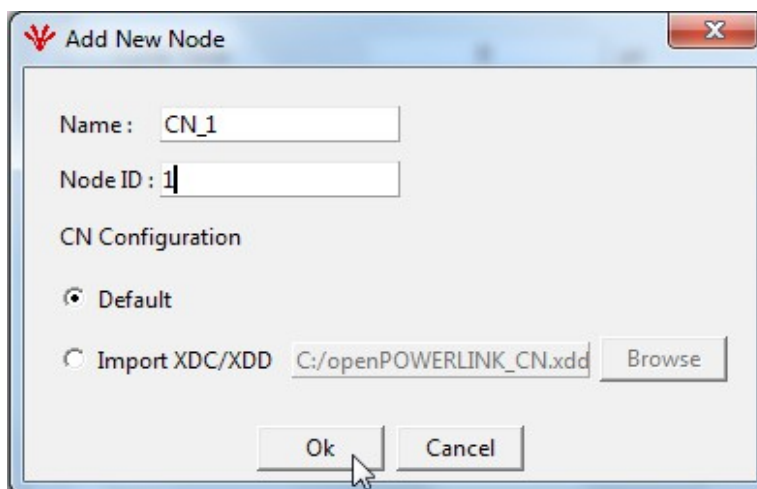


*Illustration 6: Add CN Menu*

Now the 'Add New Node' dialog box will pop prompting the user to give the required configuration for the CN.

New Node Configuration	Description
Name	Name of the Node.
Node ID	Node Id of the Node. Range(1 - 239)
CN Configuration: Default	Default CN xdd which will be available with the installation package.
CN Configuration: Import XDD / XDC	User defined configuration for the CN.

In this sample project, enter Name as 'CN\_1' and NodeID as '1' and select 'Default' CN xdd as the configuration as shown in Illustration 7: Add CN Window.

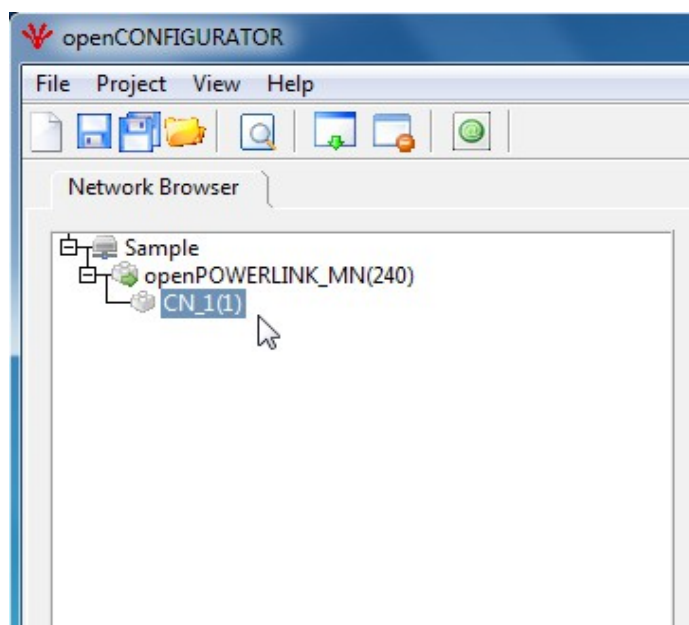


*Illustration 7: Add CN Window*



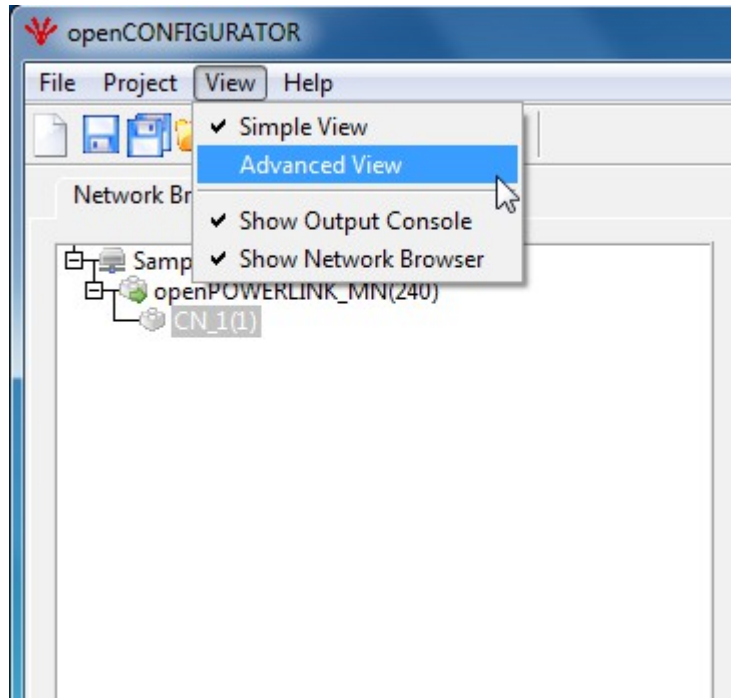
**Note:** If you choose to import your XDD, Please validate your XDD with the XDD-Check tool (a free utility available at the EPSG homepage: <http://www.ethernet-powerlink.org>) before importing it into openCONFIGURATOR.

After adding a CN the project window will look similar to Illustration 8: CN Created.



*Illustration 8: CN Created*

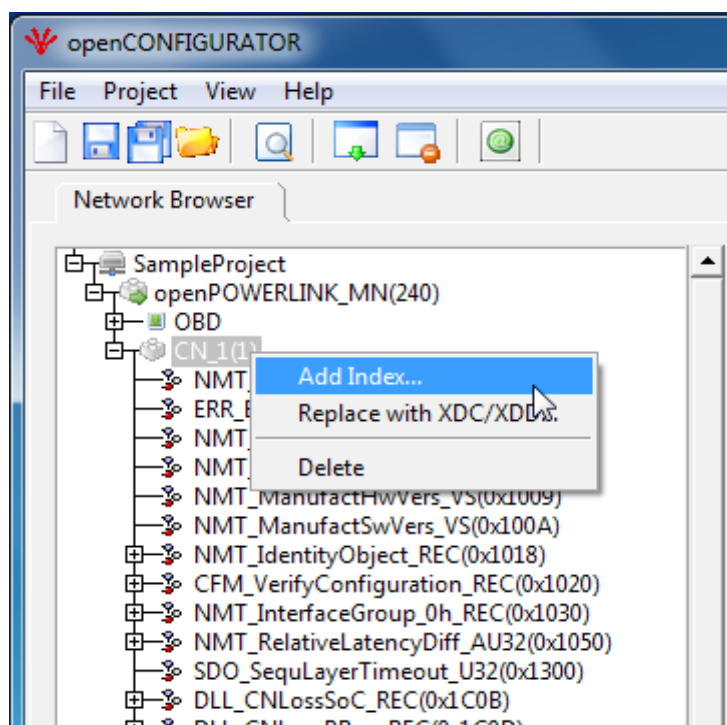
Select View > Advanced View as shown in Illustration 9: View Menu. This is required to list the available index and PDO mapping details of the CN.



*Illustration 9: View Menu*

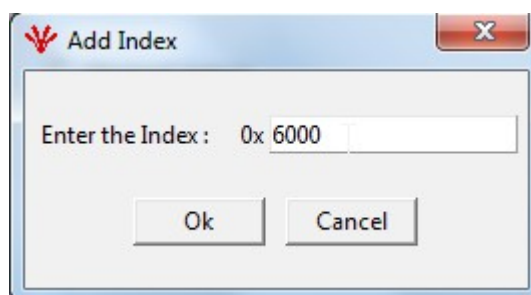
### 3.3 Adding process variable to openPOWERLINK CN

To add an output process variable index to a CN, right click on the CN and select 'Add Index' option as shown in Illustration 10: Add Index Menu.



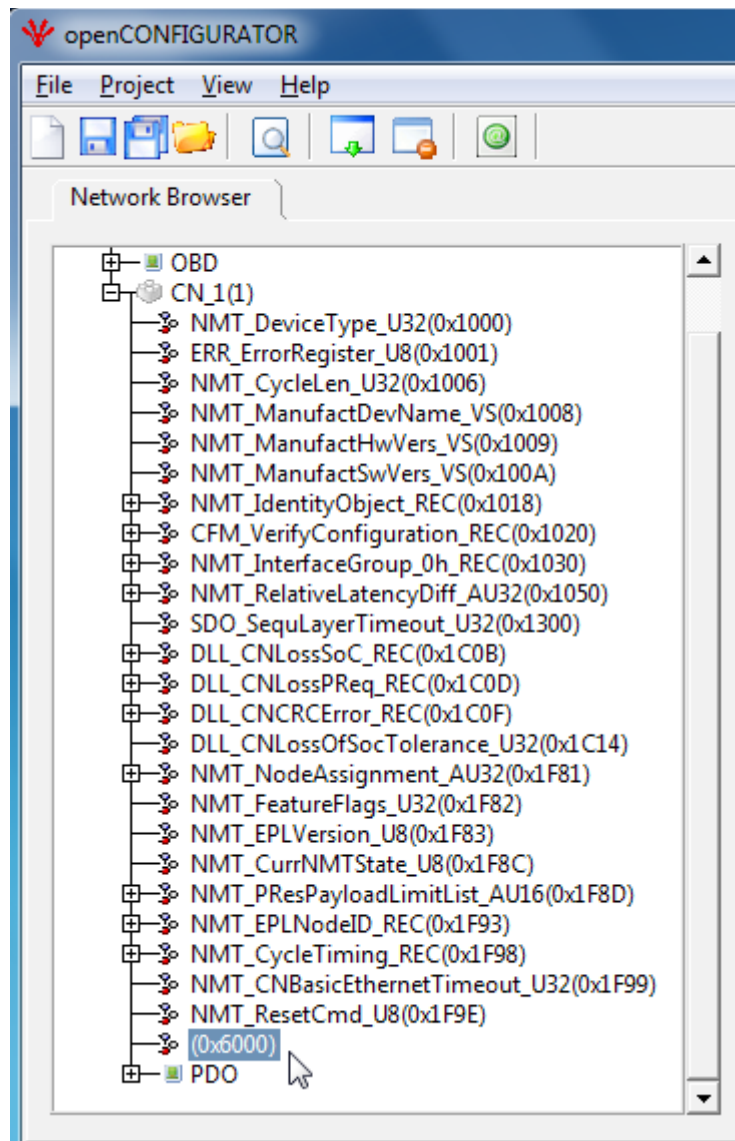
*Illustration 10: Add Index Menu*

The 'Add Index' pop up window will appear asking for Index Id. Give the Index Id in hexadecimal (say 0x6000) and click 'Ok' to add the index as shown in Illustration 11: Add Index Window.



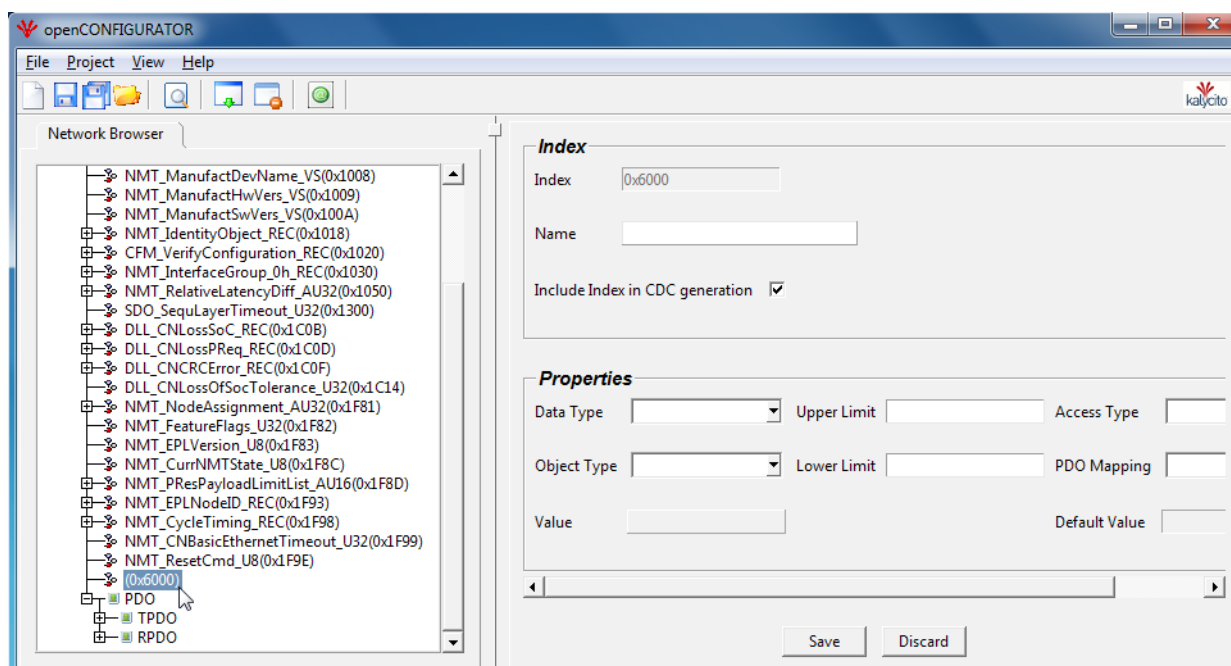
*Illustration 11: Add Index Window*

The Index (0x6000) will be added to the Node(CN\_1) as shown in Illustration 12: Index Added - Tree.



*Illustration 12: Index Added - Tree*

Click on this newly added Index(0x6000) to display its 'Properties' on the right pane as shown in Illustration 13: Index Properties - Empty.



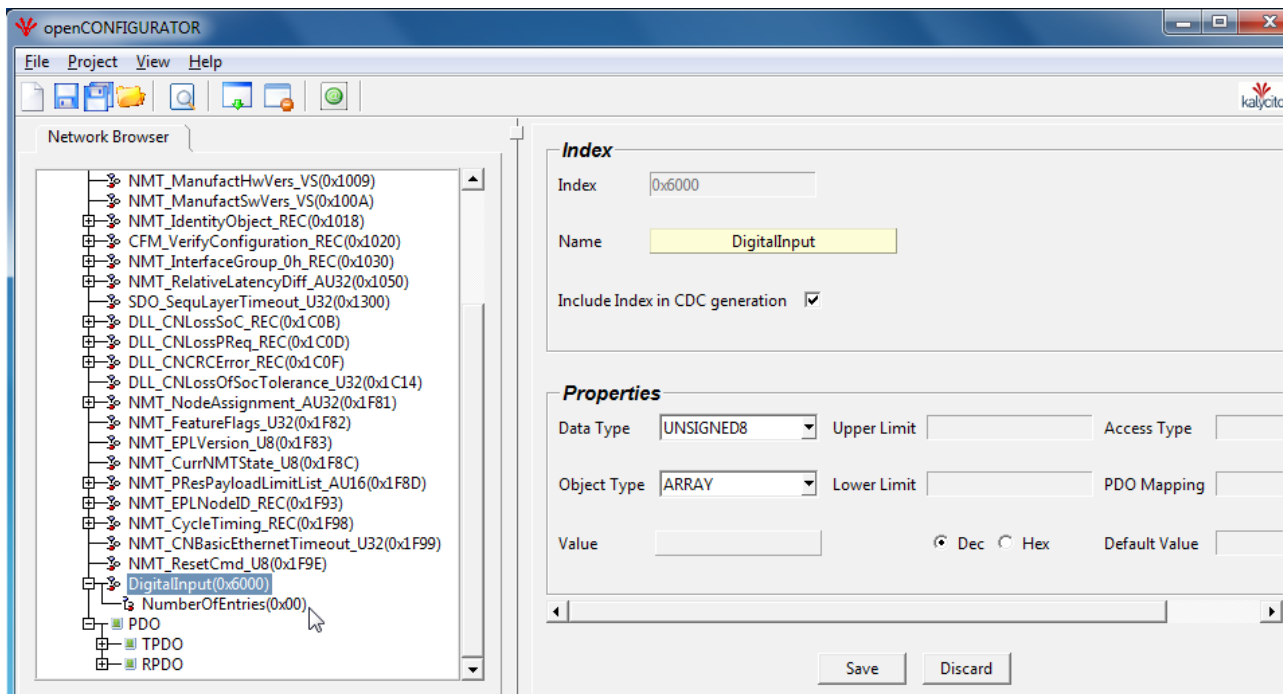
*Illustration 13: Index Properties - Empty*

Property	Description
Index*	Index Id.
Name*	Name of the Index / SubIndex.
Include In CDC generation	Determines the inclusion of the value of the Index in the CDC.
DataType*	Data type of the object.
ObjectType*	Object type of the object.
AccessType*	Access rights for the particular object.
PDO Mapping*	Determines the mapping type for the PDO object.
Default Value	Default Value for the object.
Value(Actual Value)	Actual Value for the object.
Dec/Hex	Toggle between the decimal and hexadecimal view of the value.
Upper Limit	Highest limit for the value of the object.
Lower Limit	Lowest limit for the value of the object.

\* mandatory fields

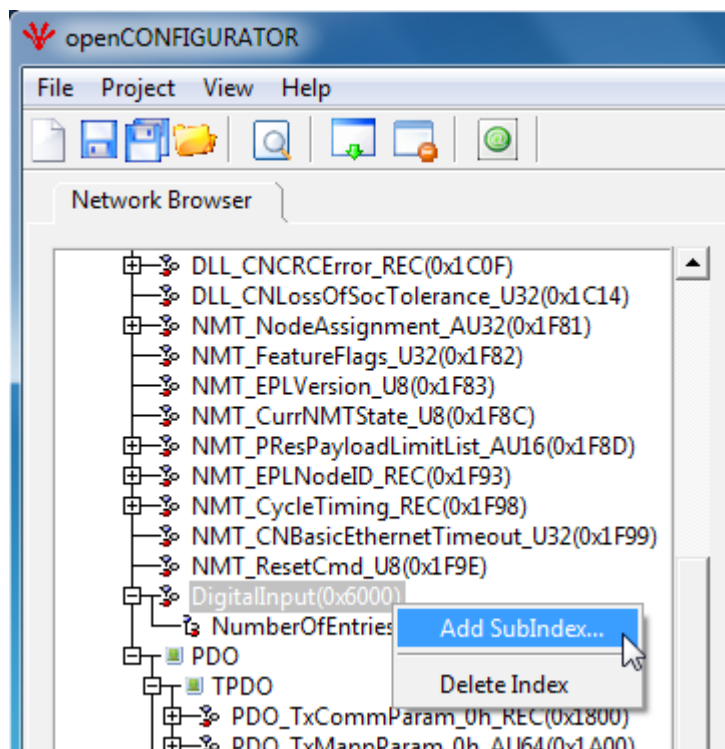


Enter the 'Name' as 'Digital Input', select 'the 'Object Type' as 'Array' and 'DataType' as 'UNSIGNED8' and click 'Save'. The Index properties will be saved and the SubIndex with SubIndexId(0x00) will be created automatically as shown in Illustration 14: Index Added.



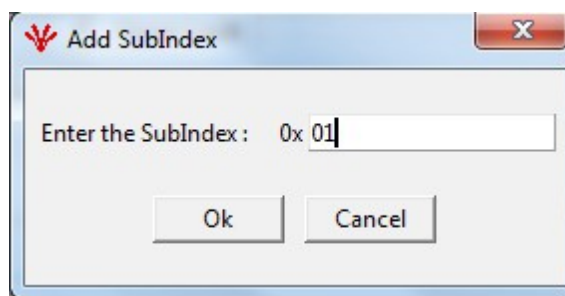
*Illustration 14: Index Added*

To add a SubIndex for the output process variable, right click on the process variable index (0x6000) and click on 'Add SubIndex' option in the sub menu as shown in Illustration 15: Add SubIndex Menu.



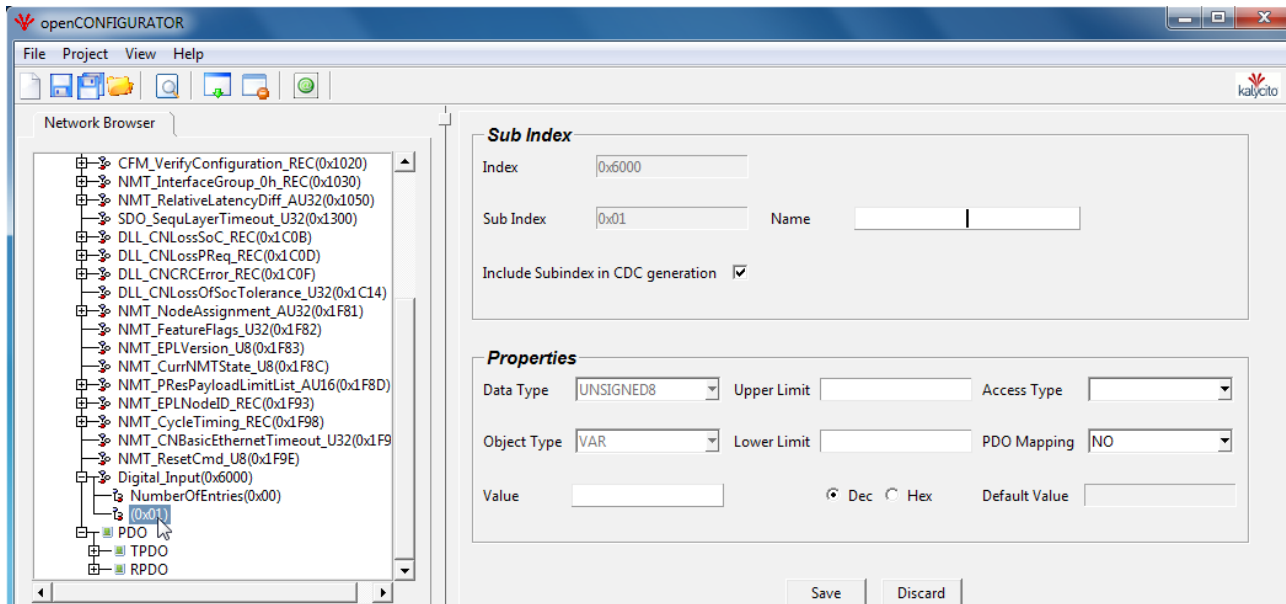
*Illustration 15: Add SubIndex Menu*

A small pop up window opens asking for SubIndexId. Give the SubIndexId in hex(say 0x01) and press 'Ok' to add the SubIndex as shown in Illustration 16: Add SubIndex Window.



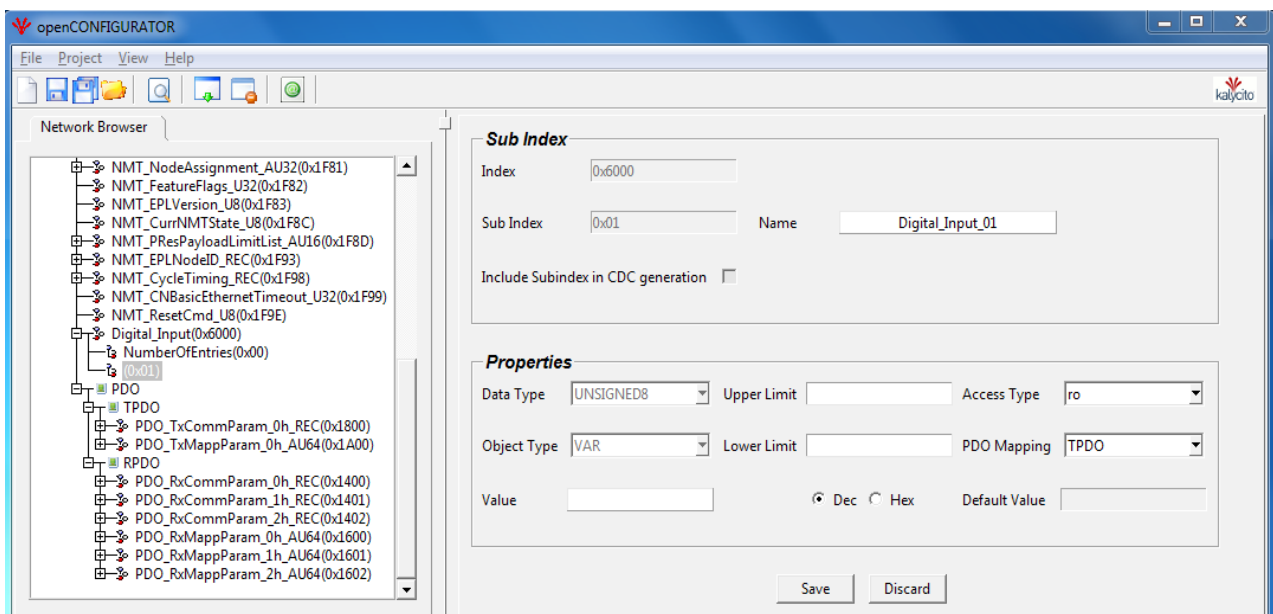
*Illustration 16: Add SubIndex Window*

Then click on the newly added SubIndex (0x01) under the Digital\_Input (0x6000). The fields under 'Properties' tab will be empty as shown in Illustration 17: Add SubIndex Properties- 1.



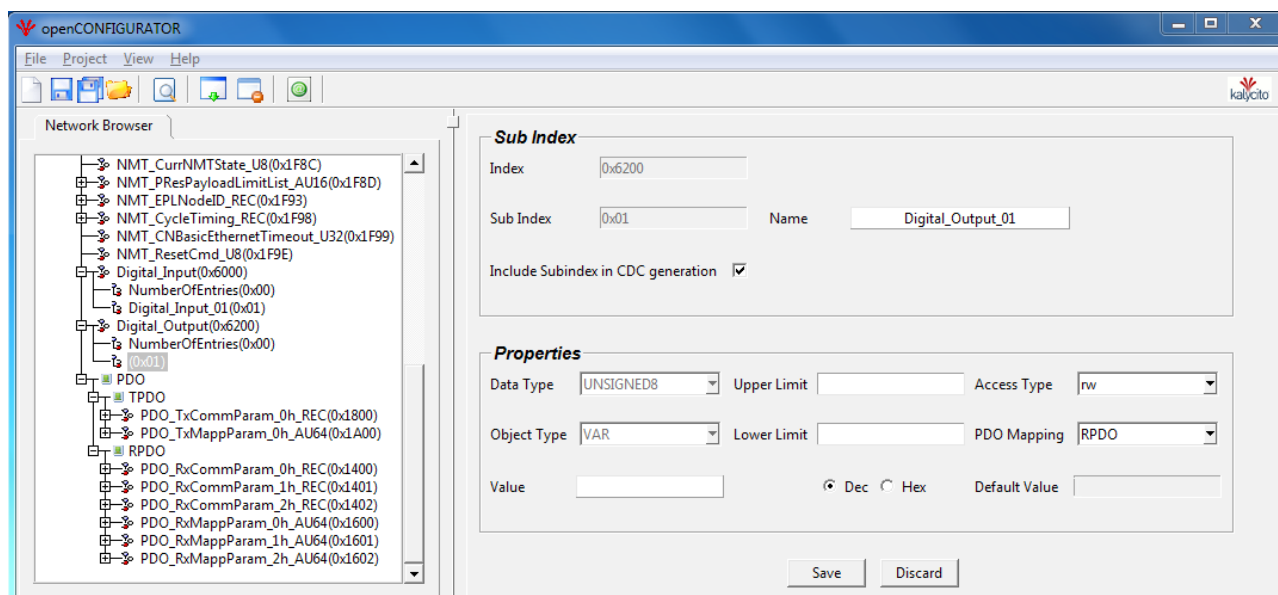
*Illustration 17: Add SubIndex Properties- 1*

Then enter/select the fields under 'Properties' tab on the right pane and click 'Save' as shown in Illustration 18: Add SubIndex Properties- 2



*Illustration 18: Add SubIndex Properties- 2*

Similarly an input process variable can be added with IndexId(0x6200) and SubIndexId(0x01) as shown in Illustration 19: Input PI variables.

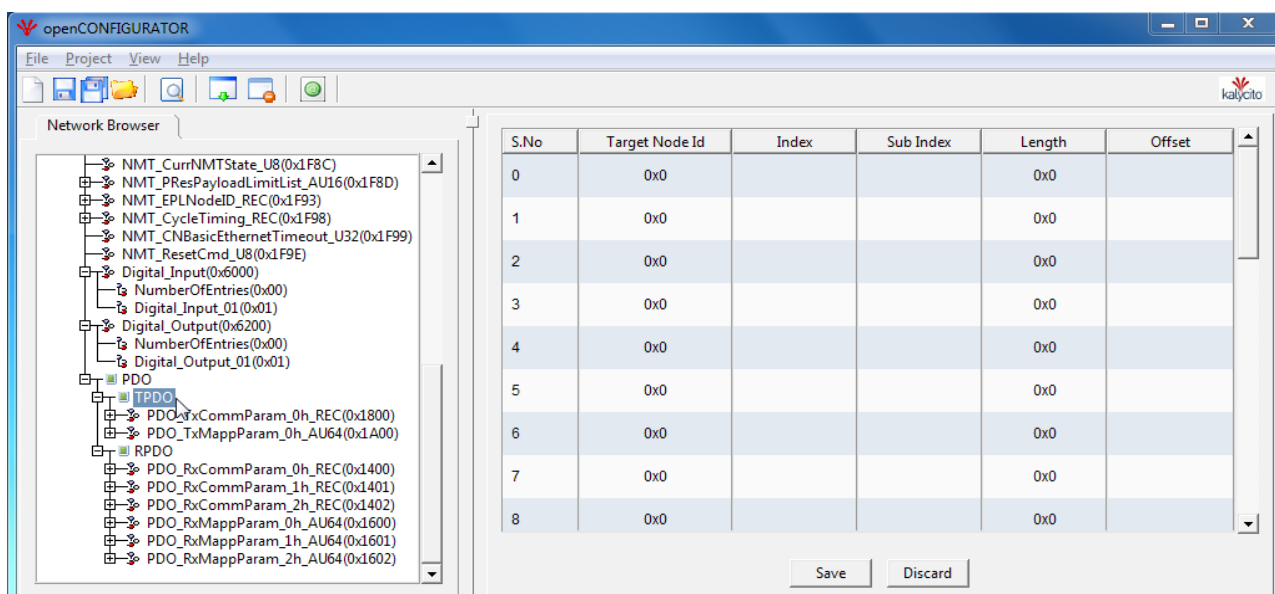


*Illustration 19: Input PI variables*

### 3.4 PDO Mapping for the CN process variables

A PDO(Process Data Object) is used to exchange process variables between the managing node(MN) and the controlled nodes(CN) of the POWERLINK network. The two types of PDO are Receive PDO (RPDO) and Transmit PDO (TPDO). (For more information, refer section 6.4 of Ethernet POWERLINK Communication Profile Specification Version 1.1.0).

Click on TPDO from the “Tree Browser”. A PDO mapping table will be loaded on the right pane as shown in Illustration 20: PDO mapping table.



S.No	Target Node Id	Index	Sub Index	Length	Offset
0	0x0			0x0	
1	0x0			0x0	
2	0x0			0x0	
3	0x0			0x0	
4	0x0			0x0	
5	0x0			0x0	
6	0x0			0x0	
7	0x0			0x0	
8	0x0			0x0	

*Illustration 20: PDO mapping table*

Select the appropriate 'Target Node Id' as shown in Illustration 21: PDO mapping table - Select Node id.

S.No ▲	Target Node Id	Index	Sub Index	Length	Offset
0	0x0			0x0	
1	0x0 0xF0 0x1			0x0	
2	0x0			0x0	
3	0x0			0x0	

*Illustration 21: PDO mapping table - Select Node id*

Select an 'Index' value available in that list as shown in Illustration 22: PDO mapping table - Select Index Id.

S.No ▲	Target Node Id	Index	Sub Index	Length	Offset
0	0x0			0x0	
1	0x0	0x6000 0x6200 0x0000		0x0	
2	0x0			0x0	
3	0x0			0x0	

*Illustration 22: PDO mapping table - Select Index Id*

Select a 'Sub-index' value within the range(0x00 – 0xFE) as shown in 12Illustration 23: PDO mapping table - Select SubIndex Id.

S.No ▲	Target Node Id	Index	Sub Index	Length	Offset
0	0x0	0x6000	0x	0x	
1	0x0		0x01 0x00	0x0	
2	0x0			0x0	

*Illustration 23: PDO mapping table - Select SubIndex Id*

Select the 'Length' of the PDO object as shown in Illustration 24: PDO mapping table - Select Length.

S.No ▲	Target Node Id	Index	Sub Index	Length	Offset
0	0x0	0x6000	0x01	0x	
1	0x0			0x0008	
2	0x0			0x0000	
3	0x0			0x0	

*Illustration 24: PDO mapping table - Select Length*

The 'Offset' values will be updated automatically after the 'Length' is selected as shown in Illustration 25: PDO mapping table - Offset.

S.No ▲	Target Node Id	Index	Sub Index	Length	Offset
0	0x0	0x6000	0x01	0x0008	0x0000
1	0x0			0x0	0x0008
2	0x0			0x0	0x0008
3	0x0			0x0	0x0008

*Illustration 25: PDO mapping table - Offset*

Save the changes by clicking on the 'Save' button or discard the changes by clicking on the 'Discard' button.

Similarly, create the PDO mapping for the input process variable with the Index (0x6200 - Digital\_Output), SubIndex (0x01 - Digital\_Output\_01) with 'Length' as 0x0008 as shown in Illustration 26: PDO mapping table - RPDO mapping values.

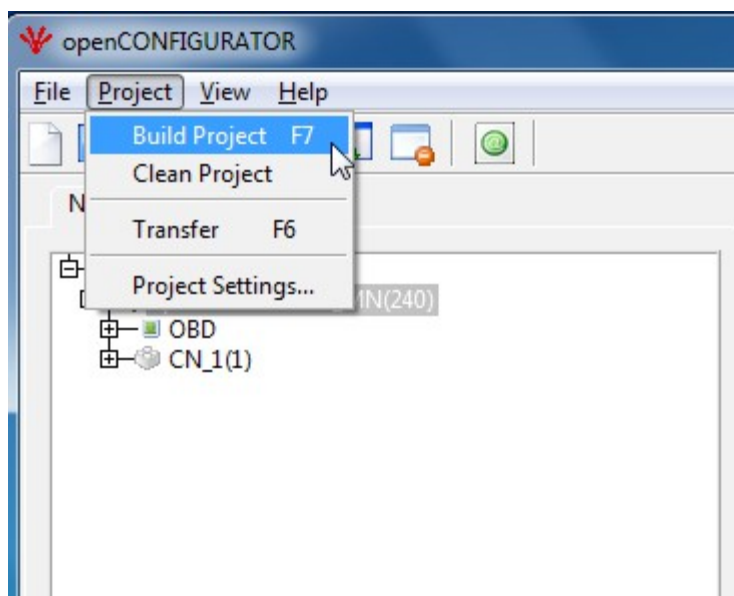
S.No	Target Node Id	Index	Sub Index	Length	Offset
0	0x0	0x6200	0x01	0x0008	0x0000
1	0x0			0x0	
2	0x0			0x0	
3	0x0			0x0	
4	0x0			0x0	
5	0x0			0x0	
6	0x0			0x0	
7	0x0			0x0	
8	0x0			0x0	

*Illustration 26: PDO mapping table - RPDO mapping values*



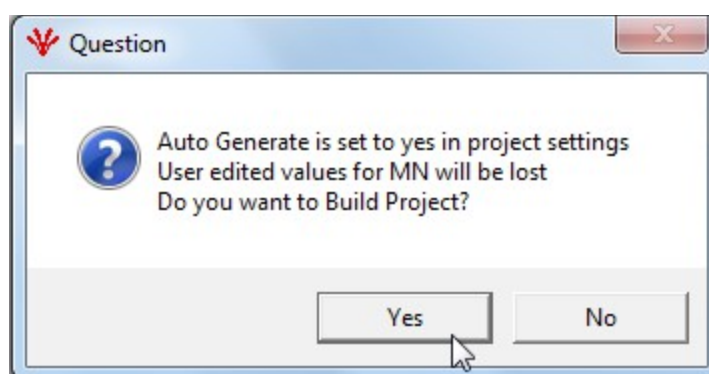
### 3.5 Build the sample project

The user can build the project by selecting Project > Build Project as shown in Illustration 27: Build Project Menu or by using the keyboard shortcut 'F7' or by clicking on the build project icon.








*Illustration 27: Build Project Menu*

After selecting the 'Build Project' option, a message pop-up will indicate that the user edited values for MN will be lost since PDO mapping will be calculated automatically. Click 'Yes' as shown in Illustration 28: Build Project - Auto Generatesince Auto Generate MN OBD is enabled. This will automatically generate MN PDO mapping.



*Illustration 28: Build Project - Auto Generate*

The files listed in Illustration 29: cdc\_xap Folder View will be created after the project is built successfully. These files will be present in the absolute path <Project location >/<Project Name>/cdc\_xap folder.

Name	Type	Size
 mnobd.cdc	CDC File	1 KB
 mnobd.txt	Text Document	1 KB
 ProcessImage.cs	CS File	1 KB
 xap.h	H File	1 KB
 xap.xml	XML Document	1 KB

*Illustration 29: cdc\_xap Folder View*

File name	Description
mnobd.cdc	CDC binary file used with the openPOWERLINK stack
mnobd.txt	Text version of the binary CDC file
XAP.h	Header file for the application
XAP.xml	XML file with the variables names, Datatype, Datasize, ByteOffsets, BitOffsets
ProcessImage.cs	A C# namespace with the application variables and the size of the data

**mnobd.txt** will be generated as below

00000012

//// NodId Assignment

1F81	01	00000004	00000007
1600	00	00000001	00
1A00	00	00000001	00
1006	00	00000004	0000C350
1C02	01	00000004	00000028
1C02	03	00000004	00000028
1C09	01	00000004	00000028
1F26	01	00000004	00002A1E
1F27	01	00000004	04034C48
1F92	01	00000004	00006978
1400	01	00000001	01
1600	01	00000008	000800000001A4C0
1800	01	00000001	01

```

1A00 01      00000008      0008000000001A040
1600 00      00000001      01
1A00 00      00000001      01
////Configuration Data for CN-1
1F22 01      00000079
0000000B
1600 00      00000001      00
1A00 00      00000001      00
1006 00      00000004      0000C350
1020 01      00000004      00002A1E
1020 02      00000004      04034C48
1C0B 03      00000004      00000050
1C0D 03      00000004      00000050
1600 01      00000008      00080000000016200
1A00 01      00000008      00080000000016000
1600 00      00000001      01
1A00 00      00000001      01
//// NodeId Reassignment
1F81 01      00000004      80000007

```

**XAP.h** will be generated as below

```

#ifndef XAP_h
#define XAP_h

# define COMPUTED_PI_OUT_SIZE 4
typedef struct
{
    unsigned CN1_M00_Digital_Input_Digital_Input_01:8;
    unsigned PADDING_VAR_1:24;
} PI_OUT;

# define COMPUTED_PI_IN_SIZE 4
typedef struct
{
    unsigned CN1_M00_Digital_Output_Digital_Output_01:8;
    unsigned PADDING_VAR_1:24;
} PI_IN;

```

```
#endif
```

**XAP.xml** will be generated as below

```
<?xml version="1.0" encoding="UTF-8"?>
<ApplicationProcess>
  <ProcessImage type="output" size="1">
    <Channel Name="CN1.Digital_Input.Digital_Input_01" dataType="Unsigned8" dataSize="8"
    PIOffset="0x0000" BitOffset="0x00"/>
  </ProcessImage>
  <ProcessImage type="input" size="1">
    <Channel Name="CN1.Digital_Output.Digital_Output_01" dataType="Unsigned8" dataSize="8"
    PIOffset="0x0000" BitOffset="0x00"/>
  </ProcessImage>
</ApplicationProcess>
```

**ProcessImage.cs** will be generated as below

```
using System;
using System.Runtime.InteropServices;

namespace openPOWERLINK
{
    /// <summary>
    /// Struct : ProcessImage Out
    /// </summary>
    [StructLayout(LayoutKind.Explicit, Pack = 1, Size = 4)]
    public struct AppProcessImageOut
    {
        [FieldOffset(0)]
        public byte CN1_M00_Digital_Input_Digital_Input_01;
        [FieldOffset(1)]
        public byte PADDING_VAR_1;
        [FieldOffset(2)]
        public byte PADDING_VAR_2;
        [FieldOffset(3)]
        public byte PADDING_VAR_3;
    }
}
```

```
/// <summary>
/// Struct : ProcessImage In
/// </summary>
[StructLayout(LayoutKind.Explicit, Pack = 1, Size = 4)]
public struct AppProcessImageIn
{
    [FieldOffset(0)]
    public byte CN1_M00_Digital_Output_Digital_Output_01;
    [FieldOffset(1)]
    public byte PADDING_VAR_1;
    [FieldOffset(2)]
    public byte PADDING_VAR_2;
    [FieldOffset(3)]
    public byte PADDING_VAR_3;
}
}
```

## 4 Conclusion

The project creation, configuration setup and generation of CDC, XAP & C# namespace outputs has been demonstrated for a project having a CN with one RPDO and one TPDO. These outputs can be used as an input to openPOWERLINK MN for running the application.

## 5 Appendix – Abbreviations

MN	Managing Node
CN	Controlling Node
CFM	Configuration Manager
PI	Process Image
OBD	Object Dictionary
PDO	Process Data Object
CDC	Concise Data Configuration
XDD	XML Device Description
XDC	XML Device Configuration
XAP	XML Application

## 6 Support

Please post your queries and suggestions on the appropriate topic in the openCONFIGURATOR discussion forum at Sourceforge.