NEDA STDF Loader
开发手册

A Light and Intelligent Solution

Nornion NEDA STDF Loader
开发手册

First Printing, August2019

Document Number: NL-004-01 Rev. A

目录

1 开始

NEDA STDF Loader 是什么 1-1
开发环境说明 1-2
NEDA.dll 1-3
创建 Visual Studio 项目工程 1-4

2 手册

命名空间 2-1
NEDA 类 2-2
属性和事件 2-3
方法 2-4
事件属性示例 2-5
返回数据的结构 2-6

1
开始

- **NEDA STDF Loader** 是什么.
- 开发环境说明.
- **NEDA.dll**.
- 创建 **Visual Studio** 新项目工程

NEDA STDF Loader 是什么？

NEDA STDF Loader (NEDA.dll)是一个 STDF 解析控件，可以解析指定的 STDF 文件，并把结果以机构化的数据返回主调程序。NEDASTDFLoader 支持同步和异步两种调用方式，异步调用时用户界面不会挂起，可以继续相应用户操作以获得更好的用户体验。

开发环境说明

开发语言和环境:

- Microsoft .NET 4.0 或者以上.
- Visual Studio 2017 或者更高版本.
- 支持.NET 环境下的编程语言 C#, VB, C++等.

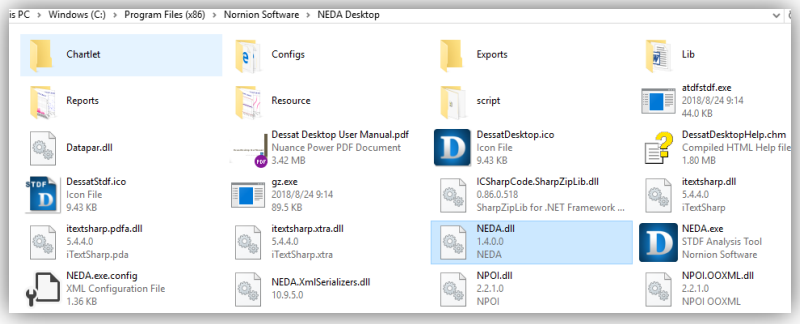
NEDA 控件环境:

- 需要安装 NEDA Desktop Edition 最新版本并具有有效授权 (可以申请 1 个月的试用授权来开发或者学习

NEDA.dll

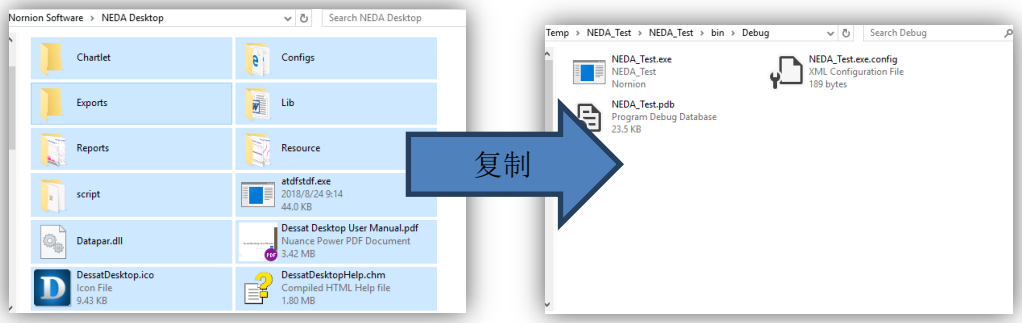
在安装好 NEDA Desktop Edition STDF 分析工具之后，你可以在安装目录下找到 NEDA.dll，这个就是我们需要的 NEDA STDF Loader 控件，我们可以用它来做二次开发，通过它从 STDF 中解析数据，并将数据存储到我们的服务器或者数据库来做更深层次的分析。

如：C://Program Files(x86)/Nornion Software/NEDA Desktop/

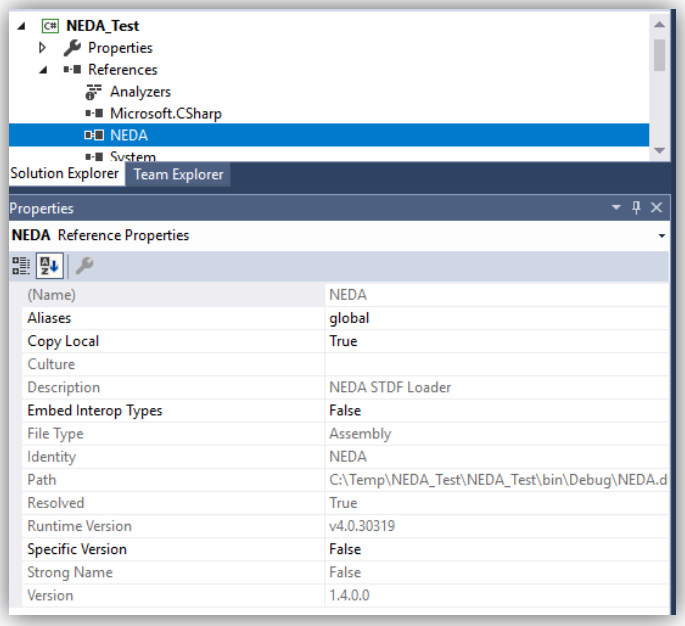


创建 Visual Studio 项目工程

创建一个 Visual Studio 新项目，并把 NEDA Desktop Edition 安装目录下的所有文件复制到新建工程的 bin/Debug 目录下。



在项目中添加对 NEDA.dll 对的引用



2
手册

- 命名空间.
- **NEDA** 类.
- 属性和事件.
- 方法
- 事件属性示例
- 返回数据结构

命名空间

引入对 NEDA.dll 的引用后，我们的 NEDA 类在 Nornion 命名空间里面。

```
Nornion.NEDAnda = newNornion.NEDA();
```

NEDA 类

其中类 NEDA 就是我们需要调用的主要的类，用 NEDA 类完成我们从 STDF 文件中解析数据的任务。

属性

Bool ErrorFlag, string ErrorMsg: 这是一个重要的 flag，每次操作之后需要检查是否被置位(ErrorFlag=true),如果被置位表示操作过程中有异常发生，错误信息保存在 ErrorMsg 中。

我们主要的操作有 2 个：初始化和解析，所以在初始化和解析之后都需要检查 ErrorFlag。

ProgressChangedEventHandlerProgressChangedEvent: 异步解析进度更新事件，在异步解析时用来指定更新界面的函数。

RunWorkerCompletedEventHandlerWorkerCompleteEvent：在异步解析完成后的回调函数

DataSetStdDataSet: 返回数据，解析完成后此 DataSet 中会存储解析出来的数据

HashtableStdLotInfo: 返回数据，解析完成后此 Hashtable 中会存储 STDF Lot 相关的信息

HashtableStdfWaferInfo:返回数据，解析完成后此 Hahshtable 中会存储 STDFWafer 相关的信息

Note: 如果 ErrorFlag=true, 返回数据无效

方法

BoolParseStdf(stringStdfFileName):解析 STDF 的方法，把 STDF 文件的绝对路径作为参数传入，解析完成后会返回是否成功的 Flag，如果成功则返回 true, 反之则返回 false。 也可以在解析完成后检查 ErrorFlag 属性来判断是否有错误发生。如果没有任何错误发生，可以从 StdfDataSet, StdfLotInfo, StdfWaferinfo 中提取解析的的数据。

voidParseStdfAsync(stringStdfFile)：开始异步解析，把 STDF 文件的绝对路径作为参数传入， 解析完成后系统会自动调用 WorkerCompleteEvent 属性指定的回调函数。在开始异步解析前，请指定 WorkerCompleteEvent 和 ProgressChangedEvent 对应的函数。异步解析完成后再提取数据之前，请检查 ErrorFlag 是否被置位。

voidDispose(): 析构函数，将数据内存交给 GCC 回收。

事件属性示例

这里简单说明事件属性 WorkerCompleteEvent 和 ProgressChangedEvent 如何赋值。

```
nda.ProgressChangedEvent = newProgressChangedEventHandler(UpdateExtractProgress);
nda.WorkerCompleteEvent =
newRunWorkerCompletedEventHandler(ExtractCompleteCallback);
nda.ParseStdfAsync(stdf_file);

publicvoidUpdateExtractProgress(object sender, ProgressChangedEventArgs e)
{
labelStatus.Text = e.ProgressPercentage.ToString() + "%";
}

publicvoidExtractCompleteCallback(object sender, RunWorkerCompletedEventArgs e)
{
labelStatus.Text = "Extraction complete!";
//检查置位标志, if return=true, 解析过程中有错误发生
if (!nda.ErrorFlag)
     {
RenderStdfResult(nda);//自定义函数把数据提取显示到界面或者存储到数据库
     }
else
     {//Some error happened during extraction, show Error msg
labelStatus.Text = nda.ErrorMsg;
     }

}
```

返回数据结构

HashtableStdfLotInfo

Type	Key
System.Char	CMOD_COD
System.Byte	STAT_NUM
System.String	AUX_FILE
System.String	LOT_ID
System.String	SUPR_NAM
System.String	TST_TEMP
System.String	OPER_FRQ
System.String	HAND_TYP
System.String	ROM_COD
System.String	LOAD_ID
System.DateTime	SETUP_T
System.Char	RTST_COD
System.String	USER_TXT
System.String	SERL_NUM
System.String	CARD_TYP
System.String	SBLOT_ID
System.DateTime	START_T
System.String	EXTR_TYP
System.String	TEST_COD
System.String	PROC_ID
System.String	USR_DESC
System.String	DIB_TYP
System.String	FLOW_ID
System.String	PKG_TYP
System.String	DATE_COD
System.UInt16	BURN_TIM
System.DateTime	FINISH_T
System.String	EXEC_TYP

HashtableWaferLotInfo

Type	Key
System.Byte	WF_UNITS
System.String	EXC_DESC
System.Single	DIE_HT
System.UInt32	PART_CNT
System.DateTime	START_T
System.UInt32	RTST_CNT
System.String	FABWF_ID
System.Single	DIE_WID
System.Int16	CENTER_X
System.String	WAFER_ID
System.String	USR_DESC
System.Byte	POS_X
System.Byte	POS_Y
System.String	FRAME_ID
System.Single	WAFER_SIZ
System.UInt32	ABRT_CNT
System.String	MASK_ID
System.UInt32	GOOD_CNT
System.Char	WF_FLAT
System.UInt32	FUNC_CNT
System.Int16	CENTER_Y
System.DateTime	FINISH_T

System.String	SPEC_NAM
System.String	DSGN_REV
System.String	CABL_ID
System.String	ENG_ID
System.String	JOB_REV
System.String	DIB_ID
System.String	EXTR_ID
System.String	HAND_ID
System.String	CONT_ID
System.String	LASR_TYP
System.Char	PROT_COD
System.String	LOAD_TYP
System.String	CARD_ID
System.String	EXC_DESC
System.String	JOB_NAM
System.String	SITE_NUM
System.String	FLOOR_ID
System.Char	DISP_COD
System.String	OPER_NAM
System.String	TSTR_TYP
System.String	SPEC_VER
System.String	PART_TYP
System.Byte	SITE_CNT
System.String	CABL_TYP
System.String	LASR_ID
System.String	FACIL_ID
System.String	SETUP_ID
System.String	FAMLY_ID
System.String	NODE_NAM
System.String	EXEC_VER
System.String	CONT_TYP
System.Char	MODE_COD

StdfDataSet 中的数据表 DataTable 的 Name

```
TestLimits -- contains definition of all parametric tests [limits and unit]
TestData   -- contains results of all parametric tests [test reading]
HBin       -- contains Hardware bin information
SBin       -- contains software bin information
TestCount  -- contains statistical count info [executed, failed...] of all tests
PartCount  --contains site execute part count info [normally it is not used]
```

我们准备一个名为“NEDA_Test”的Demo程序，演示了如何使用NEDA Loader控件解析STDF，参阅示例程序可以让你更快了解如何使用NEDA控件。