WinTool Guidelines for CAM Interfaces

17.04.2014

WinTool Guidelines for CAM Interfaces, rev 4

These guidelines shall give suggestions on how to create *WinTool* Interfaces for CAM systems that provide the user with a straight experience throughout the whole software compilation.

WinTool AG
Flüelastrasse 7
CH-8048 Zurich, Switzerland

Phone: +41 (0)44 401 00 55
http://www.wintool.com
Mail: info@wintool.com

Contents

Summary

Job of *WinTool*-CAM-Interfaces

WinTool-CAM-Interfaces enable the user to select and transfer assemblies from the *WinTool* database to the CAM environment. From the data stored in *WinTool*, the CAM Interface either generates a 3D model of the tool assembly or uses user defined 3D models and tool outlines for tool path simulation. The cutting conditions for the different work materials are transferred from the *WinTool* technology library and get attached to the tool in the CAM system. A complete list of every used tool assembly per NC-Program will be stored in the *WinTool* database for further use as setup sheet, documentation and queries.

Purpose of these Guidelines

As many different CAM systems exist, there already also exist many different interfaces for them to interact with *WinTool*. While developing Interfaces to over 15 different CAM Systems throughout the last 10 years, we, *WinTool* AG, went some different paths about project planning, programming, interaction of different software, archiving and user experience and we want to help you to create a software your customer will be happy with.
Although not everything mentioned maybe applicable to every CAM system, these guidelines shall give suggestions on how to create *WinTool* Interfaces for CAM systems that provide the user with a straight experience throughout the whole software compilation.

Contact

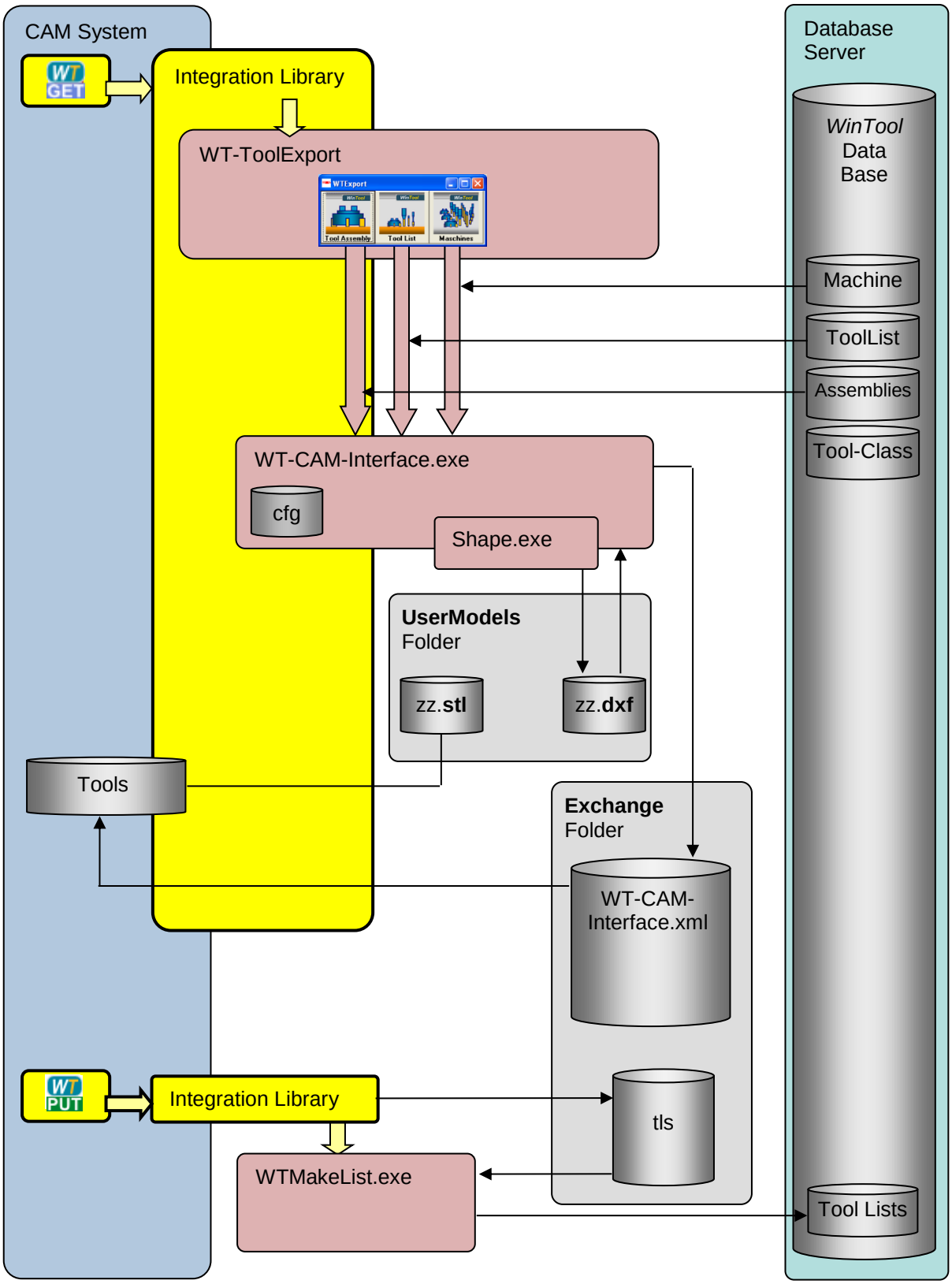
This document shall be a constant work in progress. It will not cover every aspect of *WinTool* CAM-Interface development but will give you a quick overview of some best practices. To get the most recent version or if you feel something should be changed or added, please feel free to contact

WinTool AG
Flüelastrasse 7
CH-8048 Zürich

Phone: +41-44-401 00 55
Mail: info@WinTool.com
Site: www.*WinTool*.com

Workflow, Installation and Configuration

Workflow



Workflow Explained

When receiving the CAM-Integration Package from *WinTool*, you will get the three Software modules WT-ToolExport, WT-CAM-InterfaceApp and WTMakeList. These modules care for the interaction with the *WinTool* database.
The user will start the whole process thru the "Integration Library" (symbolized by the "GET" icon). This will start WT-ToolExport, which is the "controller" application where the user can select which assemblies, he needs. WT-ToolExport will then automatically start the WT-CAM-InterfaceApp, which ensures the correct data retrieval from *WinTool* and will store that data in an XML file in the defined exchange directory, together with an outline of the tool assembly, which is created by WTxTShape.exe (installed together with *WinTool*). This data then will be imported by the integration library. The library will know that the whole data retrieval process has finished once WT-ToolExport Windows process has closed.
When the user has finished his CAM project, he will tell the integration library to write a "tls file", which can be read by WTMakeList. This will write the usage information of the assemblies in the NC Project back into *WinTool*.

Installation

As simple base rule – try to keep close to the "Windows Installer and Logo Requirements":
http://msdn.microsoft.com/en-us/library/windows/desktop/aa372825%28v=vs.85%29.aspx

Folders, Files and Rights

The user data is separated from the program files. This is in compliance with Microsoft standards.

Program Files

It may be needed for the proper function of the CAM Integration to put some files into special directories defined by the CAM system, but else, if possible you should place all program files into the main application directory which is suggested to be
`[ProgramFilesFolder]\[YourCompanyName]\WT-[CAMName]-Interface\`
Whereby `[ProgramFilesFolder]` is a default Windows Installer Property,
`[YourCompanyName]` is the name of your company and
`[CAMName]` is the name of the CAM System the Interface is for (e.g. ESPRIT, NX, etc…)

It is mandatory that a text file named "AppData" with UTF-8 encoding is placed in the main application directory. The content of this file must be the path to the directory where the user data is stored, e.g.
`[Public Documents]\WT-[CAMName]-Interface`
The WT-CAM-InterfaceApp needs this file to find the configuration files, which are located in the user data directory.

User Data Directory

The user data should be stored in a directory inside `[Public Documents]`, e.g. `[Public Documents]\WT-[CAMName]-Interface`. It contains the configuration files of the *WinTool* applications.

When installing, you must create the user data directory and inside it the subfolders "`UserModels`" and "`Exchange`" with your setup as they will be needed by the WT-CAM-Interface application and take care that those folders as well as any configuration files you may install have the correct permissions set (`[%USERDOMAIN]` / `Everyone` ➔ `Full Control`)

Tip: In the setup, read the registry value `Common Documents` in the key
`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders\`
to determine the `[Public Documents]` folder of the installed Windows version and set the content of the "AppData" file.

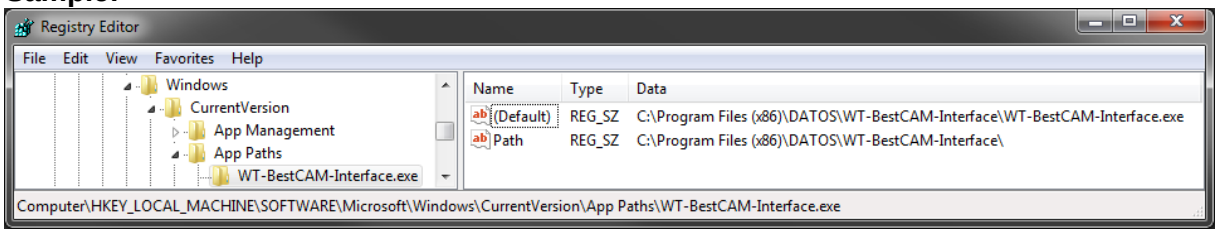
Registry

As you surely will need to know in the files that care for the integration to the CAM system where e.g. `WT-ToolExport.exe` has been installed to, you should set an "Application Path" registry entry for the file `WT-[YourCAM]-Interface.exe`, which should be installed in the main application path. This way you can later on read that registry entry to know where from to start the files that the user may have installed into a non-default directory.
The Application Path registry entries can be found in
`HKLM\Software\Microsoft\Windows\CurrentVersion\App Paths\`
Stick to the standards used by other programs here, so

- create a key with the same name as the .exe file
- fill `(Default)` with the path to the exe file (including file name)
- create a string value named "Path" filled with the path to the exe file without file name

Though you shouldn't need to create this during installation by hand, as there already are according MSI settings which you can use that will write to the corresponding entries.

Sample:



Attention!
Do not set the entry for the file `WT-ToolExport.exe` itself, as several interfaces could be installed on the customer PC which all use WT-ToolExport – so installing one of the interfaces would overwrite the registry entry for the other!

Setup Design

Here, in general also refer to the "Windows Installer and Logo Requirements" especially and in addition:

- try to keep your setup simple to handle for the user
- do not include unnecessary files
- name your features and their description, cab files and setup properly
- try to correctly handle uninstallation (remove all files except config files and licenses)
- try to correctly handle updates (correctly overwriting files, not overwriting config files) (Keep in mind Setup Version, Product and Upgrade Codes and MSI Upgrade Table)

Configuration

Try to keep your configuration centralized and as simple as possible. Preferably use as few configuration files as possible and place all of them into the main application directory.
Use standard text formats such as INI and XML and in addition you may provide a GUI to access and change the configuration values. Try to keep the configuration separate from any code, scripts and other things which are too confusing for the user and which the user might break. Also if something cannot be configured by the user, it cannot be configured wrongly.
It is suggested to add another (non writable) copy for each config file with the extension "`.default.[CfgFileExtension]`" to have a configuration file at hand that the user can copy if he messed up his real configuration.

Tool Data Calculations

T-Number Assignment

By default, the following logic is used to determine the tool assembly T-Number (ToolAssembly.TNumber):

Tool Assembly or Machine	Tool List
If the value Machines.TRelation of the machine that is assigned to the tool assembly is set to "true" and the tool assembly T-Number (Tools.T-Number) value is 0, the tool assembly T-Number is set to its identification nr (Tools. Nr). In any other case the T-Number (Tools.T-Number) is used.	The tool number in the tool list (ToolLists.T of corresponding row) for the tool assembly is used.

Before the tool assembly is added to the tool store of the CAM, make sure the determined machine place number is unique: Check the tool assemblies that are already in the tool store and if a number is already used, calculate the lowest possible free number and assign it to the new tool assembly.

CAM Integration

As the general job of *WinTool* CAM Interfaces is to GET tools from the *WinTool* tool library into the CAM system and to PUT the complete list of tool assemblies used in the NC program back into *WinTool*, 2 new Icons should be placed in a separate toolbar of the CAM system.

Will tell your integration library to start `WT-ToolExport.exe`. Once the WT-ToolExport Windows Process has finished you can start reading the generated exchange files.
(As of version 2.1 of WT-ToolExport it will report you back with a return code if the export was successful, so you won't need to try to read the exchange file if the user canceled.)

Will generate a .tls file as input for `WT-MakeList.exe`. It then will start WT-MakeList.exe with the path to the .tls file as a parameter.

Do not try to read data directly from *WinTool* other than with the CAM Integration Package supplied by *WinTool* and especially do not write any data back into *WinTool* directly. *WinTool* always should be the place where edits to tools and assemblies happen – if you write data into *WinTool* directly this may harm the customers database.
You also should always pull assemblies out of *WinTool* whenever needed and not store them for further use in the CAM systems centralized database. If a user does change tooling data in *WinTool* and the CAM programmer creating a new NC program doesn't know of this because he is using old data that was stored in the CAM's own database the documentation according with which the tool will be created and the NC program won't match which can cause broken parts, machine downtimes or even damaged machines!

Testing

During installation of *WinTool* Professional a database (WTData.mdb) with sample tool data will be installed. You can always revert to this database by simply copying that file into your *WinTool* installation directory and use this (clean) file for testing.

Note: The CAM Integration modules provided by *WinTool* use the database linked to the *WinTool* application. When you link your *WinTool* application to a different database, the Interface will also switch to that database automatically.

- Always test with the *WinTool* sample database, so you can assure that if something doesn't work on the customers site, the issue is because of the data he entered and not because of your software
- It is suggested to use the Sample Tool List "Side-Frame" to make your first tests
- STL models of the tool holders can be found in the subfolder of the *WinTool* installation
- Always document with which tool assemblies you tested the import
- Always test with the assemblies you previously used and only add new tests
- Try to keep your tests reproducible
- Maybe try to automate the testing procedure
- Always document with which version of *WinTool*, the CAM Integration Package, the CAM System and which version of your library you tested
- Only deliver fully tested software! Your customers won't enjoy banana software…

Archiving, Versioning & Documentation

Archiving

It really pays off

It may seem like a lot of pointless additional work, but it really pays off in the end, when you need to reproduce a bug someone reported to you. Always try to archive the code as well as the binaries in a way that you will later on securely be able to pick out the version you released.

Using source code repositories

Using a source code repository such as SVN, CVS, GIT or whatever you may prefer actually doesn't mean more work, but can also even *simplify* your work!
It helps you keep your software changes clean, as you always can easily compare your current work against what has been stored in the repository so you'll see if you forgot to take out that testing variable that breaks the whole code under certain conditions. In addition you can easily track your work progress thru commit messages or by using an issue tracking system that might be integrateable with your source code repository and you can always go back to a former version of your code to compare the outcome or for bug reproduction.

Versioning

Versions properly stored in the files you deliver to the customer are and should be the only reference a customer should need to determine what he actually has installed (despite the product name for sure).
In the Windows World there is the de facto standard to have Versions with the scheme
`[major].[minor].[build].[revision]`
(also see Wikipedia concerning "Versioning")
Practically you may fill those version numbers with anything you like, but – whatever you may do – please change your version number whenever you deliver a new package and never *DE*crease them in the next version!
It's a good practice to keep "major" as is (1 per default) as long as you don't do drastical changes (e.g. break compatibility to older versions), increase "minor" whenever you intend to give out a new planned release and "build" whenever you give out an unplanned or bugfixing release.
If you are using a source code repository like SVN, it makes sense to always have the "revision" field filled with the actual revision number of your project. To simplify and automate this, please find the online documentation of "SubWCRev". For other repository systems refer to the corresponding command.

Documentation

Proper documentation in combination with (as far as possible) error free coding are the most important things to consider if you want to keep your customers from constantly contacting you because the software either doesn't work, doesn't work as they think it would or because they even don't know how to use or install it.
Though this might help your documentation to be more fluently readable, never use synonyms for anything important. It will just confuse the user as he might think you're speaking of different things.
Try to cover every aspect of your software that might be of interest to the customer, but keep it short and simple as no one will want spend hours of reading for a very simple task.

History

As also done with this document, always communicate what you have changed in your software. Users won't install your new version if you don't tell them what you have changed, but they moreover will be frightened that you might have changed something that they liked so much the way it was.

Revision 4 – 14-04-17

- Updated section "Installation" due to new directory structure
- Updated section "T-Number Assignment"

Revision 3 – 12-06-20

- Added new section "Tool Data Calculations"

Revision 2 – 12-03-30

- Added Sample for "App Paths" Registry entries

Revision 1 – 12-03-27

- Initial version of this Document